VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vývoj aplikací pro Windows Phone 8

Windows Phone 8 Application Development

2014 **Jakub Holomek**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jakub Holomek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vývoj aplikací pro Windows Phone 8
Windows Phone 8 Application Development

Zásady pro vypracování:

Mobilní platformy zažívají v současné době nebývalý rozmach. Cílem práce je ilustrovat možnosti vývoje aplikací pro platformu Windows Phone 8.

1. Popište možnosti a specifika platformy Windows Phone, a to především z pohledu vývoje aplikací.
2. Navrhněte několik ukázkových aplikací, které budou ilustrovat použití určitých funkčních prvků zařízení. Např. se bude jednat o aplikace využívající tzv Live Tile, datovou konektivitu, lokalizační služby.
3. Tyto ukázkové aplikace implementujte a detailně je popište.
4. Zhodnoťte možnosti platformy Windows Phone celkově a také z pohledu efektivity a náročnosti vývoje aplikací.

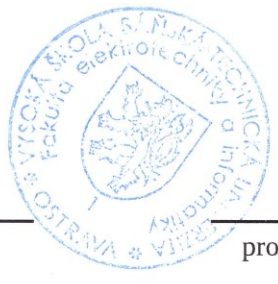
Seznam doporučené odborné literatury:

[1] M. MacDonald: Pro Silverlight 4 in C#, 2010, Apress, ISBN: 9781430229797
[2] APP HUB: http://create.msdn.com
[3] Professional Windows Phone 7 Application Development: Building Applications and Games Using Visual Studio, Silverlight, and XNA: Wrox, ISBN:978-0470891667, 2011

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2014



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 28.04.2014 ……
podpis studenta

Poděkování

Rád bych poděkoval Ing. Michalu Radeckému, Ph.D. za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Tato bakalářská práce je věnovaná vývoji mobilních aplikací pro operační systém Windows Phone 8. Popisuje specifika platformy a obsahuje několik ukázek vypracovaných aplikací s jejich rozsáhlejším popisem. Jednotlivé aplikace se zaměřují na základní funkčnosti systému, jako je ovládání pomocí gest, práce s instalovanými senzory a grafickými objekty, ukládání dat, provádění operací v pozadí a LiveTile dlaždice. Práce se také se zabývá náročností vývoje a následnou převoditelností aplikací napříč platformami.

Klíčová slova

Microsoft, Windows Phone, Silverlight, Visual Studio 2012 Express for Windows Phone, XAML, mobilní aplikace, akcelerometr, mapy, GPS, LiveTile, PCL, Flick, IsolatedStorage, ApplicationSettings

Abstract

This bachelor's thesis is dedicated to the development of mobile applications for operating system Windows Phone 8. It describes the specifics of the platform and contains several examples of developed applications with their larger decription. Individual applications are focusing on the basic functionality of the system, such as control through touch gestures, work with installed sensors and graphical objects, data storage, operations processing in background and LiveTile tiles. The work also deals with the demans of development and subsequen scalability of applications across platforms.

Key words

Microsoft, Windows Phone, Silverlight, Visual Studio 2012 Express for Windows Phone, XAML, mobile application, accelerometer, maps, GPS, LiveTile, PCL, Flick, IsolatedStorage, ApplicationSettings

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
API	Application Programming Interface	Rozhraní pro programování
GPS	Global Positioning system	Zařízení pro určování polohyNFC
NFC	Near Field Communication	Sada pro bezdrátovou komunikaci
OS	Operation System	Operační systém
PCL	Portable Class Library	Přenositelná knihovna tříd
SDK	Software Developing Kit	Softwarový balíček pro vývoj
SLAT	Second level Address Translation	Hardwarová virtualizace
WP8	Windows Phone 8	Mobilní operační systém
XAML	Extensible Application Markup Language	Značkovací jazyk grafického rozhraní

Obsah

Úvod

V dnešní době, kdy jsou naše životy ovlivněny moderními technologiemi, si dokáže málokdo představit život bez nich. Nejčastějšími zařízeními, se kterými se lidé nejvíce setkávají a dokonce nosí denně s sebou, jsou mobilní telefony nebo tablety.

Technologické procesy se neustále zdokonalují a umožňují zvyšování výkonu se sníženým odběrem energie a zároveň se zmenšením velikostí komponent, což vede k neustálému rozšiřování působností zařízení v různých oblastech lidských činností. Spojením komponent do jednoho zařízení vznikl nástroj, který dokázal nahradit desítky jiných a získat si tak právoplatně místo nejpoužívanějšího zařízení.

Významným prvkem sjednocujícím tato zařízení je operační systém, který kompletně zastřešuje funkcionalitu a umožňuje rozšiřování jejich možností. Můžeme se setkat s širokou řadou operačních systémů. Všechny systémy si ovšem nejsou rovny z pohledu procentuálního zastoupení na trhu, a proto se nejčastěji zmiňují systémy Android, iOS a nováček Window Phone.

Pokud existuje zařízení s operačním systémem, přicházejí na řadu vývojáři, aby dokázali tato zařízení co nejlépe představit běžným lidem a to z pohledu možností a schopností těchto zařízení. Pokud mají zařízení a jejich operační systémy dostatek dobrých a cenově přijatelných aplikací, tak roste jejich obliba a tím i tržní procentuální zastoupení. Čím je systém oblíbenější, tím více se vkládá úsílí do jeho dalšího rozšiřování a zlepšování pro co nejlepší uživatelský zážitek.

Tato práce bude věnována podání základní představy o vývoji na nejmladším mobilním operačním systému, a to na Windows Phone ve verzi osm. Součástí práce jsou reálné příklady, na kterých dojde k vysvětlení věnovaného tématu.

1 Popis platformy a její specifika

1.1 Historie Windows Phone

Windows Phone není prvním počinem firmy Microsoft na poli s mobilními operačními systémy, těmi byly operační systémy Pocket PC a později Windows Mobile, a to až do verze 6.5.5. [1] Tyto systémy byly ve své době velice populární (2000 - 2009), ovšem Microsoft si v posledních letech působení postupně nechal ujet vlak a konkurence se posunula mnohem rychleji kupředu a umožnila vytlačit tyto systémy z trhu s mobilními zařízeními.

Microsoft si uvědomil svou situaci a započal vyvíjet novou platformu, kterou představil světu v roce 2010 na World Mobile Congress nazvanou Windows Phone 7. [2] Podle čísla verze systému je jasné, že systém pokračuje v rodinné historii, avšak Microsoft se rozhodl, že systém nebude zpětně kompatibilní a také nebude tak otevřený jako jeho předchůdci. Problémem tohoto systému je, že přišel o několik let později, aby mohl zachytit trendovou vlnu. Systém nenabízel takové možnosti jako jeho konkurenti a proto nebyl mezi uživateli přijat velmi vřele.

Následovalo ještě několik aktualizací, než se Microsoft rozhodl vytvořit další systém založený na jiném jádře nazvaný Windows Phone 8, který ovšem neumožňoval uživatelům WP7 přejít na tuto verzi, ale umožňuje spouštět aplikace napsané pro WP7. WP8 v době psaní této práce dostal již tři aktualizace, kterými konečně funkcionálně dohnal svou konkurenci a v některých ohledech ji i předběhl.

1.2 Popis Windows Phone

Nově vytvořený systém měl být revolučním počinem z hlediska uživatelského prostředí, u kterého se snažil nevycházet ze vzhledu konkurenčních operačních systémů. To se Microsoftu povedlo a po dlouhém zkoumání a testování přišel s grafickým prostředím nazvaným Modern UI, kde jsou centrálním ovládacím prvkem dlaždice reprezentující aplikace a jejích data, které jsou seskupeny do řádků a sloupců. Vzhled a celý systém byl hlavně navržen tak, aby co nejméně vyčerpával baterii, a proto Microsoft zvolil výchozím profilem zobrazení černou barvu pozadí a světlý text se sjednocenými barevnými kombinacemi pro popředí. Při měřeních se došlo k závěru, že na stávajících technologiích displejů použitých u mobilních zařízení má černá barva nejnižší spotřebu.[3]

Obrázek 1.1: Úvodní obrazovka WP8 v několika nastaveních. Zdroj [2]

Dominantním prvkem ovládání celého systému je multidotykový kapacitní displej doplněný třemi hardwarovými tlačítky umístěnými vespod zařízení, kterým je přiřazena funkčnost zajišťující hladké a jednoduché ovládání celého systému. Levé tlačítko v podobě šipky slouží pro návrat zpět. Prostřední tlačítko s logem systémů Windows slouží pro návrat na úvodní obrazovku. Poslední tlačítko reprezentuje lupu pro vyhledávání. Funkcionalita tlačítek pro přechod zpět a vyhledávání je ovlivněna různými typy aplikací.

Celý systém je rozdělen do několika Hubů (sekcí), kde každý z nich reprezentuje určité funkce. Hub je pro kontakty a účty lidí, textové a MMS zprávy, E-mailového klienta, kalendář, fotky, hudbu + videa, platby, hry, práci s dokumenty studia Office.

Velkou snahu také společnost věnovala integrování sociálních sítí přímo do systému, aby měl uživatel funkční propojení se svými účty a mohl pomocí minimální snahy zasílat emaily, fotografie, či informace o svých činnostech přímo do určených aplikací. Provázanost je systémová, tudíž se s účty propojují i uživatelská data (kontakty, poznámky, fotografie, atd.). Microsoft také samozřejmě dbal na co největší využití a propojení se svými vlastními produkty a službami jako jsou Office, Exchange, Bing, XBOX Live, Internet Explorer a dalších.

Pokud by uživateli nevyhovovala funkcionalita systému či aplikace, může navštívit vestavěný Store a zakoupit si novou aplikaci. Aplikace neustále přibývají, takže se uživatel nemusí bát, že by nějaké řešení, které vyžaduje, nebylo do budoucna ke stažení. Windows Phone ovšem nepovoluje instalovat aplikace odjinud než z oficiálního Store.

Co se týče hardwarové výbavy, Windows Phone umožňuje svým zařízením být stejně vybavené jako jakékoliv jiné konkurenční zařízení a navíc jako první přinesl možnost bezkontaktních plateb a přenosů dat pomocí NFC čipu.

1.3 Zařízení Windows Phone 8

Microsoft nabízí svůj mobilní operační systém mnohým výrobcům mobilních zařízení, ovšem pod podmínkou licencování s omezeními. Výrobce musí zaplatit za licenci na jedno zařízení 10 – 15 dolarů a také musí splňovat hardwarové nároky. [4] Požadavky na hardware neboli Chassis jsou následující:

- kapacitní displej, rozlišení WVGA 480 x 800px, WXGA 1280 x 768px, 720p 1280 x 720px, Full HD 1920 x 1080 px, podpora minimálně čtyř dotyků zároveň
- minimálně dvoujádrové procesory Qualcomm Snapdragon s architekturou ARM
- minimálně 512 MB RAM pro WVGA a 1 GB pro WXGA/720p
- minimálně 4 GB flash paměti
- GPS, A-GPS, akcelerometr, světelný senzor, senzor vzdálenosti, vibrační motorek (nepovinně kompas, gyroskop, NFC a FM rádio)
- Podpora micro-USB 2.0
- 3.5 mm jack, náhlavní stereo sluchátka s 3 tlačítky pro ovládání
- zadní kamera minimálně VGA rozlišení s LED nebo Xenon přisvětlením, volitelná přední kamera
- 802.11 b/g a Bluetooth (802.11 n/ac volitelně)
- Grafický hardware podporující DirectX s hardwarovou akcelerací pro Direct3D
- Tlačítka: zpět, start, hledání, zapnutí / uspání, ovládání hlasitosti a případně dvoupolohové tlačítko pro ovládání fotoaparátu

Některé požadavky se oproti předchozí verzi snížili, jako třeba fotoaparát. V případě WP7 byl vyžadován čip s minimálním rozlišením pět megapixelů. Ovšem cena výsledných zařízení společně s licencemi byla nastavena poměrně vysoko a nedostatečně konkurovala, a proto se Microsoft uchýlil ke kroku snížení požadavků. Výrobci ovšem stejně ví, že zařízení s minimálními hardwarovými specifikacemi by uživatele stejně nezaujala, a tedy se vydávají cestou nejlepších hardwarových výbav, snižování vlastních marží nebo vydáváním několika modelů s několika měsíčním odstupem. Těmto zařízením se instaluje nejnovější dostupný hardware, aby se dřívější verze zlevnily a byly cenově dostupnější.

2 Základ pro vývoj na platformě

Microsoft stanovil nejen hardwarové parametry pro zařízení, které budou disponovat operačním systémem Windows Phone, ale určil také povinné hardwarové a softwarové parametry stanic, na kterých budou vývojáři chtít psát své aplikace.

2.1 Hardwarové nároky

Co se týče hardwarových nároků, procesor zařízení, na kterém se bude vyvíjet, musí podporovat *Hyper-V* [5] a *SLAT* technologii. [6] *Hyper-V* je vyžadována pro běh virtuálních nástrojů, konkrétně pro emulátor běhu systému Windows Phone 8 a *SLAT* se využívá pro překlad adres virtuálního zařízení na fyzické adresy. Zjistit, zdali je zařízení vyhovující, jde pomocí utility *Coreinfo*. Co se týče ostatních hardwarových součástek zařízení, je požadováno alespoň 4GB RAM, pro co nejplynulejší zpracovávání a minimálně 6 GB volného místa na pevném disku z důvodu instalace potřebných nástrojů pro vývoj.

Reálné zařízení při vývoji již není potřeba. Emulátor je velice sofistikovaný a plně simuluje funkčnost běžného zařízení s konkrétním rozhraním. Samozřejmě emulátor plně nenahrazuje testování na reálném zařízení, a to z důvodu mnoha hardwarových kombinací a ovladačů jim přidružených.

2.2 Softwarové nároky

Windows Phone 8 vyžaduje instalaci 64bitového stolního systému Windows 8 Pro či vyšší verzi systému. V nižších verzích nebude emulátor fungovat. Podmínkou pro plnohodnotnou funkčnost je, že systém nesmí být spouštěn skrze virtuální nástroj.

Windows Phone 8 SDK obsahuje vše potřebné pro vývoj aplikací. Balíček obsahuje samostatně fungující vývojové studio Visual Studio Express 2012 for Window Phone (SDK může být nainstalováno i na plnohodnotnou verzi Visual Studia 2012). Dále obsahuje mnohé grafické nástroje, programové knihovny a samozřejmě emulátor.

Přímo od vývojářů systému Windows Phone také existuje volně šiřitelné rozšíření funkcionality standartního SDK, které se nazývá *Windows Phone Toolkit*. Tento balíček obsahuje novou funkčnost spolu s novými komponentami, pro efektivnější vývoj aplikace.

2.3 Jiné nároky

Ohledně publikování a následného profitování na vytvořených aplikacích, je potřebné získat *Windows Live ID* na stránkách Microsoft. [7] Jakmile vývojář má přiřazeno *Live ID*, musí se pod ním zaregistrovat a vytvořit si vývojářský účet s možností vystavení aplikace na *Store*. Tato služba platí na rok a je zdarma pouze pro studenty díky programu *DreamSpark*, jinak je zpoplatněna zvlášť pro jednotlivce a pro firmy (19 a 99 $).

Pokud se vývojář rozhodne testovat a ladit své aplikace sám na reálných zařízeních, tak je potřeba každé zařízení propojit s jeho vývojářským účtem. Na jednom vývojářském účtu je možno odemknout 3 zařízení a na každé z nich nainstalovat 10 neoficiálních aplikací.

Další možností testování aplikace, kdy se vývojář nemusí osobně starat o testování aplikace je ta, že vystaví svou aplikaci na testovací verzi obchodu. Aplikace se zpřístupní tisícům testerů, kteří danou aplikaci mohou testovat a zapisovat své poznámky a hodnocení.

Posledním a nejdůležitějším požadavkem na vývojáře je základní znalost konkrétních kombinací jazyků a technologií z výpisu:

- **.NET (C#, Visual Basic)** – základní pilíře celého vývoje WP
- **C++** - omezen pouze na tvorbu her a logiky, jelikož není varianta XAML s C++
- **Silverlight** – platforma pro vývoj business a multimediálních aplikací. Používá se v kombinaci s XAML jazykem, pro aplikace využívajících základních ovládacích prvků systému
- **XAML –** založený na značkovacím jazyce XML, který je určen pro tvorbu grafického rozhraní v aplikacích firmy Microsoft
- **XNA** – platforma určená pro vytváření her nebo aplikací s 3D grafikou v jazyce C# nebo Visual Basic. Microsoft se snaží již tuto platformu vytlačit a nahradit ji technologií *DirectX*
- **HTML5 + Javascript** – pouze v případě, kdy je v aplikaci vložen webový prohlížeč

2.3.1 Publikování aplikace

Dojde-li vývojář k závěru, že aplikace je již připravena na vydání, přichází na řadu samotný proces publikace. Pod svým LiveID se přihlásí na svůj účet a zvolí položku Publish. Objeví se několika krokový postup s pevně danými pravidly a podmínkami, jakými jsou přesně daná rozlišení a formáty obrázků. Pokud se vyvojáři podáři těmito kroky projít a splnil veškeré náležitosti, tak dochází k samotnému odeslání aplikace na prošetření aplikace samotné společnosti Microsoft. Zde technici prozkoumají bezpečnost a splnění kritérii na Windows Phone aplikaci a dají v řádech hodin až dnů vědět vývojáři o rozhodnutí. Pokud aplikace uspěje, dochází k jejímu uveřejnění na Store a v opačném případě získává vývojář vyrozumnění s důvodem rozhodnutí a s dokumentem tipů a rad pro další pokus. Kompletní přehled podmínek pro publikování aplikace je uveden v této práci v části příloh.

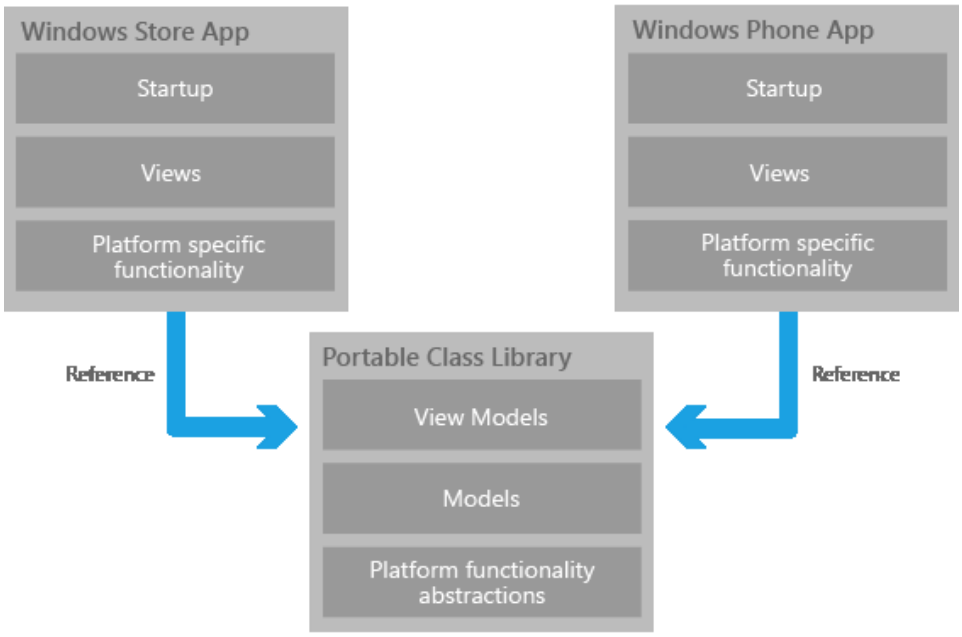
2.4 Vztah stolní a mobilní verze Windows

Windows Phone 8 je založen na jádru *Windows 8 NT Kernel*, tudíž stolní a mobilní verze systému mají společné základní jádro. K určitým komponentům, jako k souborovému systému, multimédiím, sítím a mnoha dalším se přistupuje pomocí shodných ovladačů, a tím šetří práci výrobcům a vývojářům. Další výhodou je možnost zvýšení osazení operačních pamětí, přidání podpory vícejádrových procesorů a mnoha rozlišení displejů do budoucna.

Společná podmnožina *.NET engine* a *Windows Runtime API* funkčností, akorát navíc přidány funkce specifikované pro mobilní zařízení (rozeznávání hlasu, uzamykací obrazovka, dotyková gesta atd.). Ovládací prvky uživatelského rozhraní mají ve většině případů shodný vzhled i funkčnost, ovšem v každém systému jsou kladeny jiné nároky pro co nejlepší uživatelský zážitek. [8]

2.4.1 Portable Class Library

Jedná se o knihovnu, do které umístíme společnou funkčnost pro všechna zařízení využívajících platformu *.NET*. *PCL* je doporučený způsob vývoje aplikací, pokud se plánuje vydání aplikace jak pro stolní, tak pro mobilní zařízení a ušetřit mnoho času vývoje. [9]



Obrázek 2.1: Princip Portable Class Library. Zdroj [9]

2.5 Životní cyklus aplikace

Launching – spouští-li uživatel aplikaci, ať už ze seznamu aplikací nebo z úvodní obrazovky, zavádí tím novou instanci aplikace do stavu Launching. V tomto stavu by se měla vykonávat minimální činnost, pro dojem svižnosti systému. Při prvním spuštění aplikace zde může být provedena činnost, která uživateli dává najevo, že vstupuje do aplikace (například ikona indikující načítání aplikace)

Running – po spuštění aplikace přechází aplikace do stavu Running, tedy běžného režimu funkčnosti. Zde zůstává do doby, než uživatel přejde do jiné aplikace nebo neuzamkne obrazovku.

onNavigatedFrom – metoda, která se spouští kdykoliv uživatel přechází z jakékoliv aplikace do Vaši aplikace, ale také při přechodu z aplikace ven. Tento stav může nastat i uvnitř aplikace v rámci navigace mezi stránkami aplikace. Vždy, když je tato metoda volána, měl by se uložit stav stránky pro případ, že by se uživatel vrátil zpět, ale aplikace by již byla odstraněna z paměti.

onNavigatedTo – metoda je volána pokaždé, pokud se přechází do aplikace nebo pohybuje v rámci její obsahu. V této metodě se má kontrolovat, zdali dochází k prvnímu spuštění.

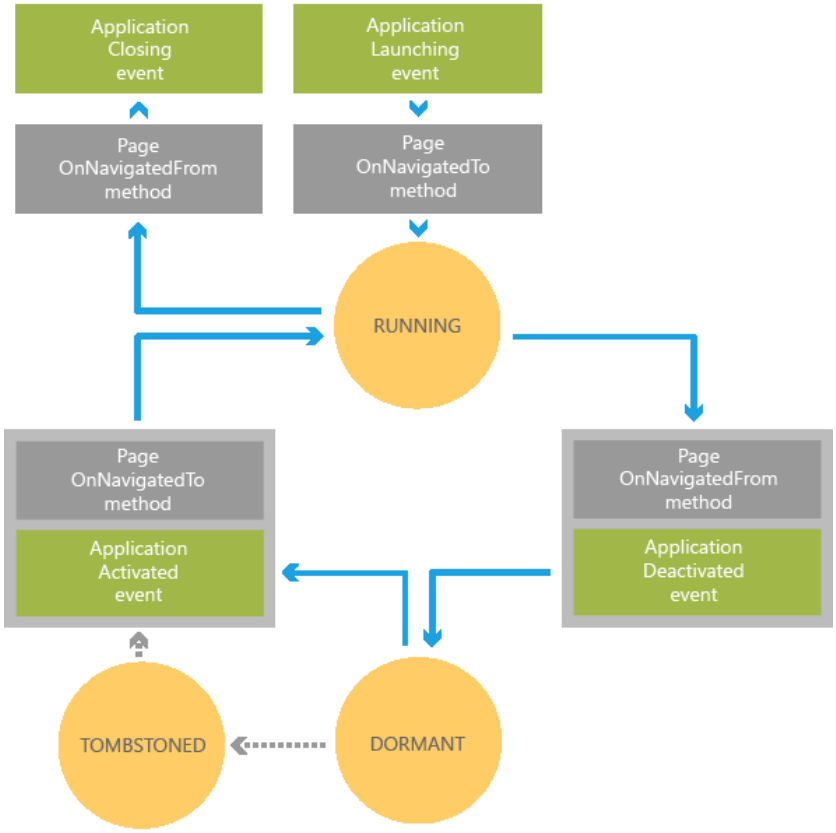
Deactivated- tato událost je vyvolána tehdy, pokud uživatel použije pro návrat zpět tlačítko start, spuštěním jiné aplikace nebo uzamknutím přístroje. V případě této události se mají zálohovat neuložená data aplikace

Dormant – jakmile je u aplikace vyvolána událost Deactivated a uživatel přechází mimo aplikaci, systém se pokusí převést aplikaci do Dormant stavu neboli spánku. Aplikace je stále v paměti, tudíž při její reaktivaci není potřeba načítat předchozí stav aplikace.

Tombstoned – pokud byla aplikace ukončena, systém si stále uchovává informace o přechodech vykonané v dané aplikaci. Systém uchovává takto pouze 5 aplikací.

Activated – tato událost nastává, když se přechází do aplikace ve stavu dormant nebo tombstoned. Pro zjištění stavu se využívá property *IsApplicationInstancePreserved* a to proto, abychom věděli, jestli musíme načíst původní stav aplikace nebo ne.

Closing – pokud uživatel z úvodní stránky aplikace přejde tlačítkem zpět pryč z aplikace, dochází k ukončení aplikace bez jakéhokoliv uložení stavu. V události *Closing* se mohou uložit uživatelská data, ale pouze po dobu deseti vteřin, než dojde k úplnému ukončení a uvolnění paměti.



Obrázek 2.2: Životní cyklus aplikace. Zdroj [10]

2.6 Background Agents

V systému Windows Phone není implementován plnohodnotný multitasking. V jeden okamžik může běžet v popředí pouze jedna aplikace, ale pro co nejlepší uživatelský zážitek a úsporu energie, dovoluje systém ostatním aplikacím využívat metody pro činnost v pozadí, takzvané Background agenty. [11] Příkladem jejich použití může být stahování dat, hudba v pozadí, streamování audia a videa, notifikace.

Každá aplikace může mít jednoho agenta. Ten může být registrován jako:

- **PeriodicTask –** vykonává se krátkou dobu (25 sekund) po opakující se interval (30 minut) a používá se pro určení polohy nebo synchronizace malého množství dat. Může být využíván vždy i při malé kapacitě baterie. Pokud ovšem uživatel nemá nastaven režim úspory energie, kdy činnost agenta nebude vykonána
- **ResourceIntensiveTask –** zaměřen na velký přesun dat, kdy zařízení není aktivně využíváno. Může běžet pouze tehdy, pokud je telefon v napájení nebo stav baterie je na více než 90 % baterie a má přístup k internetu pouze skrze Wi-Fi
- **Kombinace obou**

Každý z těchto agentů může být ukončen z důvodu překročení paměťových limitů nebo přiřazené doby vykonávání. Periodické jsou omezeny 20 MB při 1 GB paměti a 11MB při menších velikostech. Intenzivní jsou zase omezeny 10 minutami vykonávání. Počet registrovaných agentů na zařízení je omezen na 6. Pokud jakýkoliv z agentů nebude spuštěn po dobu 14 dnů, dojde automaticky k jeho deaktivaci.

2.7 LiveTile

Tile, neboli dlaždice jsou zástupci reprezentující aplikaci na úvodní obrazovce systému, která umožňuje zobrazovat informace vykonávané aplikací. Dlaždice se poprvé objevily ve Windows Phone 7, a to z důvodu lepší ovladatelnosti zařízení dotykem. Toto uživatelské rozhraní založené na dlaždicích se nazývá Modern UI. Tento styl zobrazování a ovládání se přenesl i na stolní verzi Windows. [12]

Rozeznáváme dva druhy základních dlaždic:

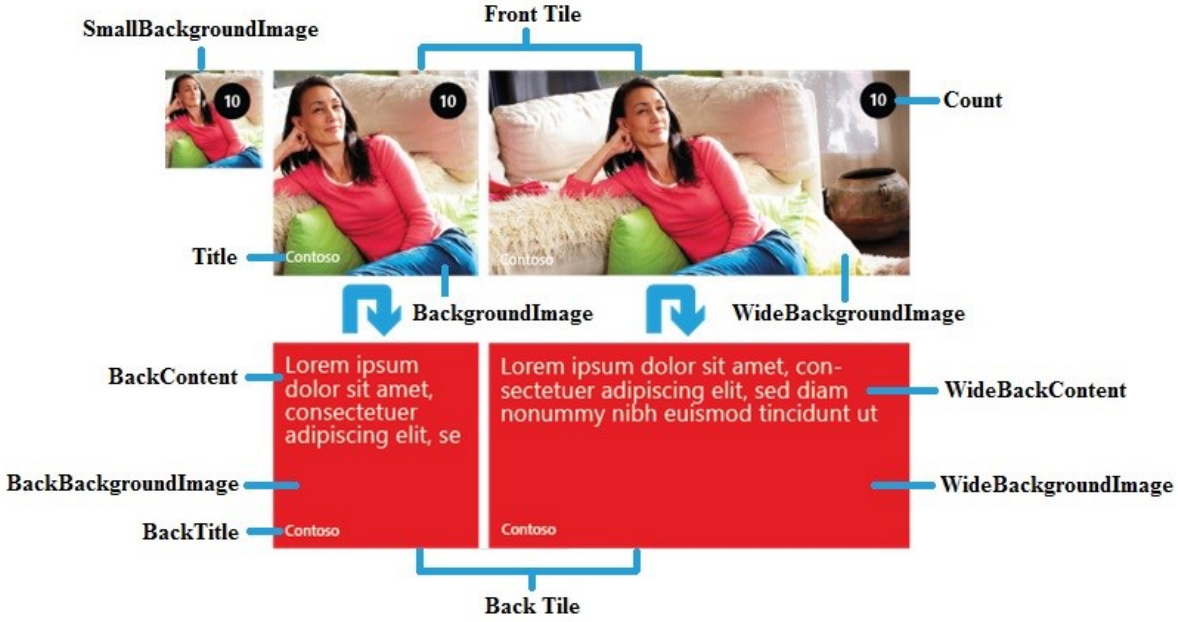
Statické dlaždice (Tile) – reprezentující pouze jeden neměnný obrázek

Aktivní dlaždice (LiveTile) – soubor obrázků měnících se po jistém časovém intervalu pomocí grafický přechodů. Mohou obsahovat data. Aktivní dlaždice nabízejí tři vizuální šablony (Flip, Iconic, Cycle) a také tři základní velikosti (malá, střední, široká).

a. Iconic šablona – dlaždice splňující Windows Phone grafické standardy. Stručně řečeno použití ikon místo obrázků umístěných na jednobarevném pozadí

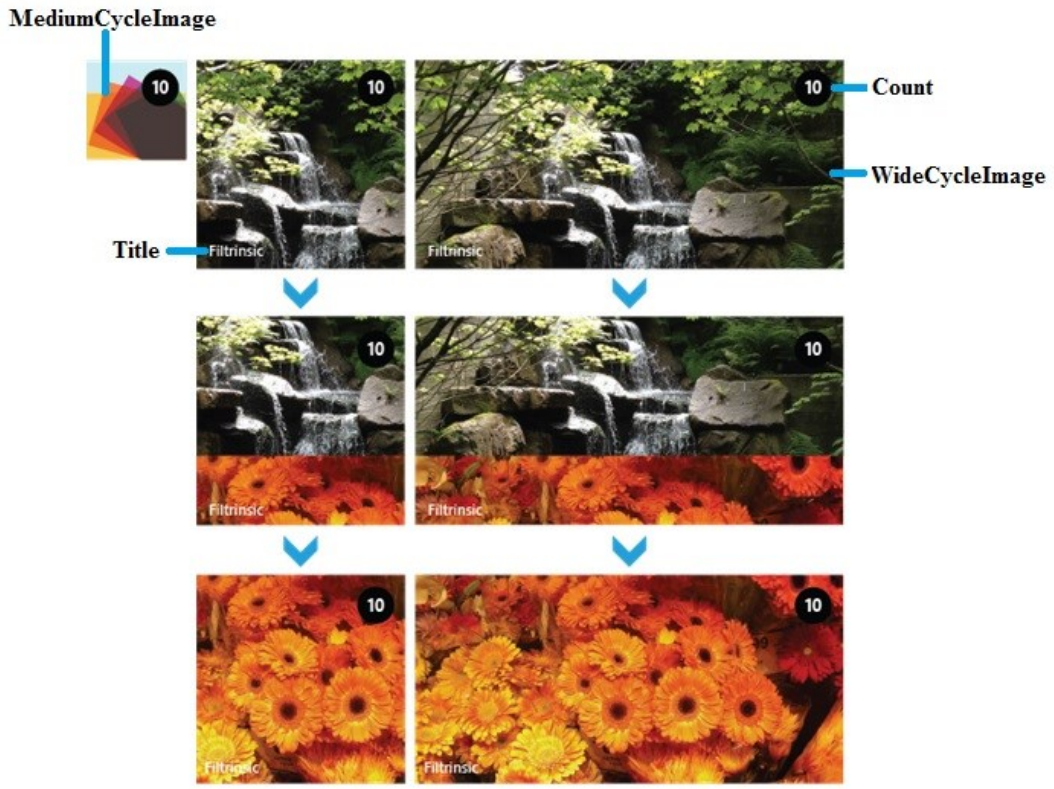
Obrázek 2.3: Iconic dlaždice s atributy. Zdroj [12]

b. Flip šablona – přetočení dlaždice ze přední na zadní stranu



Obrázek 2.4: Flip dlaždice s atributy. Zdroj [12]

c. **Cycle šablona** – dlaždice, která cyklicky prochází mezi 9 obrázky



Obrázek 2.5: Cycle dlaždice s atributy. Zdroj [12]

Dalším důležitým aspektem při návrhu dlaždic je dodržování velikostí pro podporovaná rozlišení displejů. Windows Phone 8 podporuje čtyři základní rozlišení, a to WVGA (800 x 480 px), WXGA (1280 x 768 px), 720p (1280 x 720 px) a s posledním updatem na verzi 8.1 i 1080p (1920 x 1080 px).

Tabulka 2.1: Podporované rozlišení dlaždic. Zdroj [12]

Dlaždice	Flip a Cycle [px]	Iconic [px]
Malá	159 x 159	110 x 110
Střední	336 x 336	202 x 202
Široká	691 x 336	N/A

Musí se také dodržovat formáty obrázků. Všechny obrázky musí mít formát PNG nebo JPG. Doporučuje se ovšem použití pouze formátu PNG, a to z důvodů využití průhlednosti. Obrázek pro dlaždici může být načten i z webového serveru, ale nesmí se načítat pomocí protokolu HTTPS, dále velikost obrázku nesmí být větší než 80 KB a doba stažení nesmí přesáhnout 30 sekund. Pokud obrázek načítáme z lokálního úložiště, nejsme velikostí omezeni.

2.7.1 Data na dlaždici

Data zobrazovaná na dlaždici můžeme aktualizovat pomocí tříd *ShellTile*, *ShellTileSchedule* nebo pomocí *Push Notification*. [12]

2.8 Datové zdroje a ukládání stavů

Důležitou součástí každého vývoje je ukládání, načítání a správa dat potřebných k činnosti. Pro tuto činnost nabízí Silverlight souborový systém zvaný IsolatedStorage, který umožňuje ukládat jak data, uživatelská nastavení, tak atributy aplikace.

2.8.1 IsolatedStorage

Zajišťuje virtuální souborový systém, který umožňuje zapisovat data do malých, systémem alokovaných míst na paměťovém médiu. Pro každou aplikaci instalovanou uživatelem pracující s *IsolatedStorage* je přiřazen vlastní jedinečný blok v paměti o velikosti 1 MB, který se dá požadavkem rozšířit.

Práce s *IsolatedStorage* je jednoduchá, protože se s ním pracuje, jako s běžnými datovými toky (*StreamWriter*/*Reader*, *BinaryWriter*/*Reader*) a navíc umožňuje vytvářet novou souborovou hierarchii. Pro získání požadovaného paměťového prostoru v *IsolatedStorage* musíme vytvořit objekt třídy *IsolatedStorageFile* a zavolat statickou metodu pro zajištění místa *GetUserStoreForApplication*. [13]

Zdrojový kód 2.1: Zápis do IsolatedStorage

```
try
{
      using (IsolatedStorageFile store =
      IsolatedStorageFile.GetUserStoreForApplication())
      {
            using (IsolatedStorageFileStream stream =
            store.CreateFile("datumy.txt"))
            {
                  StreamWriter writer = new StreamWriter(stream);
                  writer.Write(DateTime.Now);
                  writer.Close();
            }
            labelStatus.Text = "Datum byla zapsána do datumy.txt";
      }
}
catch (Exception err)
{
      // Chyba se vyskytne v případě zápisu do souboru, který neexistuje
      labelStatus.Text = err.Message;
}
```

Tabulka 2.2: Metody IsolatedStorage. Zdroj [13]

Metoda	Popis
CreateDirectory(), DeleteDirectory()	Vytváří novou složku, maže danou složku
CreateFile(), DeleteFile()	Vytváří nový soubor, maže soubor
Remove()	Vymaže celou přiřazenou IsolatedStorage i se soubory a složkami
OpenFile()	Otevře požadovaný soubor a vrací IsolatedStorageFileStream objekt pro další manipulaci
FileExists(), DirectoryExists()	Vrací true nebo false v závislosti na existenci objektu
GetFileNames()	Vrací pole znaků, každé pro jeden soubor v kořenovém adresáři
GetDirectoryNames()	Vrací pole znaků, každé pro složku a jejich podsložky v kořenovém adresáři

2.8.2 ApplicationSettings

Dalším typem využití *IsolatedStorage* je ukládání uživatelských dat (jméno, heslo, nastavení aplikace). K tomuto účelu vznikla třída *IsolatedStorageSetting*, která implementuje kolekci typu slovník, tudíž obsahuje klíč s hodnotou pro snadnější manipulaci. [13]

Zdrojový kód 2.2: Práce se záznamem ApplicationSettings

```
// Uložení uživatelského jména pod klíčem UserName a hodnotou
IsolatedStorageSettings.ApplicationSettings["UserName"] = “Nick”;

// Načtení uživatelského jména
string userName =
IsolatedStorageSettings.ApplicationSettings["UserName"].ToString();
```

3 Krokoměr

Krokoměr je technické zařízení zaznamenávající počet ušlých kroků. Díky zabudovanému senzoru, zvanému akcelerometr, je možné tuto aplikaci vytvořit, a tím naprosto nahradit stávající kapesní krokoměry a dále přispět k pokračování centralizovanosti mobilních zařízení.

Aplikace se bude sestávat z úvodní obrazovky, kde se uživateli budou zobrazovat měřené údaje (ušlé kroky, vzdálenost v kilometrech i mílích a počet spálených kalorií). Dále ze stránky obsahující instrukce pro správné ovládání aplikace a stránce nastavení, kde si uživatel nastaví své fyzické proporce pro co nejpřesnější výpočet.

3.1 Princip akcelerometru

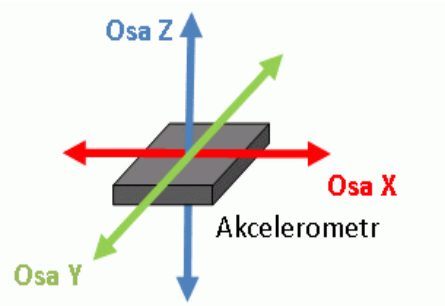
Akcelerometr neboli polohové čidlo je jeden z nejdůležitějších senzorů dnešních kapesních a přenosových zařízení. Jeho využití je nepřeberné, ať už použití ve hrách, aplikacích, či u zařízení, kde je potřeba určovat polohu v prostoru. Z fyzikálního hlediska měří akcelerometr zrychlení, které převádí na elektrický signál. Poloha je reprezentována třemi prostorovými vektory dimenze (x, y, z) určující směr a magnitudu. Magnituda se dá určit pomocí Pythagorovy věty:

$$M = \sqrt{x^2 + y^2 + z^2} \tag{3.1}$$

Magnituda je 1 (přibližné zrychlení 9.8ms^{-2}), tedy 1G běžného gravitačního působení Země na povrchu, kdy materiál, na který síla působí, stojí na místě. Při pohybu se velikost působení sil mění.

Příklady vektorů kdy zařízení zaujímá stálé místo:

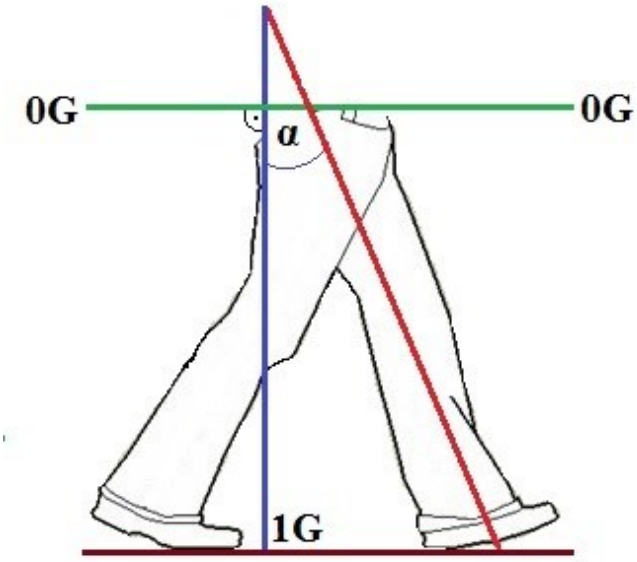
(0,-1, 0) – zařízení stojí vertikálně nahoru
(-1, 0, 0) – zařízení leží na levém boku
(0, 1, 0) – zařízení je vertikálně vzhůru nohama
(1, 0, 0) – zařízení leží na pravém boku
(0, 0,-1) – výchozí pozice kdy telefon leží na zádech
(0, 0, 1) – zařízení leží na displeji



3.2 Rozpoznání kroku

Gravitace jednoho G působí na těleso stojící nehybně na místě pod úhlem 90 stupňů se zemí. Při návrhu aplikace jsem vycházel tedy z toho, že zařízení bude při měření vloženo v kapse kalhot svisle dolů. Díky tomuto poznatku jsem si mohl určit počáteční souřadnice, od kterých se bude při výpočtu začínat.

Při každém pohybu nohy vpřed či dozadu dojde ke zrychlení, které se musí zaznamenat a porovnat s předchozí hodnotou a usoudit, ve které poloze se krok vyskytl. K tomu je určeno porovnávání hodnot magnitud (starých, aktuálních). Pozorováním a testováním náhodných rozsahů jsem došel k závěru, že ke kroku dochází, pokud se výsledná hodnota pohybuje v rozmezí 0,835 až 0,865 G.



Obrázek 3.1: Úhel rozpoznání kroku

3.1 Použití akcelerometru

Prvním krokem pro příjem údajů měření z akcelerometru je přidání reference na knihovnu obsahující práci a přístup k sensorům.

Microsoft.Devices.Sensors

Dalším krokem je vytvoření instance objektu *Accelerometer*. Pro získávání dat ze sensoru se používá několik možných přístupů.

Tabulka 3.1: Přístupy ke čtení z akcelerometru. Zdroj [14]

Přístup	Popis
GetCurrentReading	získává momentální čtení hodnot z akcelerometru
ReadingsChanged	událost vyskytující se při každé změně údajů akcelerometru
CurrentValueChanged	předává údaje z měření při každé změně hodnot

Zdrojový kód 3.1: Instance akcelererometru a zaregistrování metody pro senzorické čtení

```
using Microsoft.Devices.Sensors

public MainPage()
{
	…
	accelerometer = new Accelerometer();

	//zaregistrování metody delegátu
	accelerometer.CurrentValueChanged+=accelerometer_CurrentValueChanged;
	accelerometer.Start();
	…
}
void accelerometer_CurrentValueChanged(object sender,
SensorReadingEventArgs<AccelerometerReading> e)
{
	//Práce se senzorovým čtením
}
```

Zdrojový kód 3.2: Algoritmus rozeznání kroku

```
void accelerometer_CurrentValueChanged(object sender,
SensorReadingEventArgs<AccelerometerReading> e)
{
	float xPuvodni = 0;
	float yPuvodni = -1;
	float zPuvodni = 0;

	//Hodnoty zrychlení pro jednotlivé osy
	float x = e.SensorReading.Acceleration.X;
	float y = e.SensorReading.Acceleration.Y;
	float z = e.SensorReading.Acceleration.Z;
	//Výpočet ušlého kroku v závislosti na velikosti magnitudy
	double staraHodnota = ((xPuvodni * x) + (yPuvodni * y)) + (zPuvodni * z);
	double staraMagnituda = (double) Math.Abs((Math.Sqrt
	(Math.Pow(xPuvodni,2) + Math.Pow(yPuvodni, 2) + Math.Pow(zPuvodni, 2))));
	double novaMagnituda = (double) Math.Abs((Math.Sqrt(Math.Pow(x, 2) +
	Math.Pow(y, 2) + Math.Pow(z, 2))));

	staraHodnota /= staraMagnituda * novaMagnituda;
	if ((staraHodnota < 0.865) && (staraHodnota >= 0.835))
	{
			if (!zmena)
			{
				zmena = true;
				Settings.StepsCount.Value += 2;
			}
			else  { zmena = false; }
	}

	xPuvodni = x;
	yPuvodni = y;
	zPuvodni = z;
}
```

3.2 První spuštění aplikace

V aplikaci se dají pokrýt i případy, kdy se do aplikace přistupuje přímo ze systému, či jiné aplikace a naopak. V případě této aplikace se pokrývá případ prvního spuštění aplikace, kdy dochází k zobrazení úvodní hlášky, která uživatele vybízí k nastavení tělesných parametrů.

Zdrojový kód 3.3: Metoda pro pokrytí případu prvního spuštění aplikace

```
protected override void
OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
      base.OnNavigatedTo(e);

      if (Settings.FirstRun.Value)
      {
            this.FirstRunPanel.Visibility = Visibility.Visible;
            Settings.FirstRun.Value = false;
      }
      else
      {
            this.FirstRunPanel.Visibility = Visibility.Collapsed;
      }

      //Služba umožňující, že u přístroje nedojde k uzamčení
      PhoneApplicationService.Current.UserIdleDetectionMode =
      IdleDetectionMode.Disabled;
}
```

Obrázek 3.2: První spuštění aplikace s hláškou

3.3 LiveTile

V tomto příkladu jsem použil standartní dvoustrannou dlaždici, protože u aplikací typu krokoměr není potřeba neustále zobrazovat aktualizované data. Uživatel má zařízení většinu času vloženo v kapse a výsledek kontroluje až po výkonu.

Přední strana dlaždice obsahuje obrázek, který co nejlépe vystihuje aplikaci. Na zadní straně se zobrazují aktuální hodnoty aplikace (počet ušlých kroků, kilometrů a počet spálených kalorií). Zadní strana také slouží pro zobrazení dat z posledního výkonu.

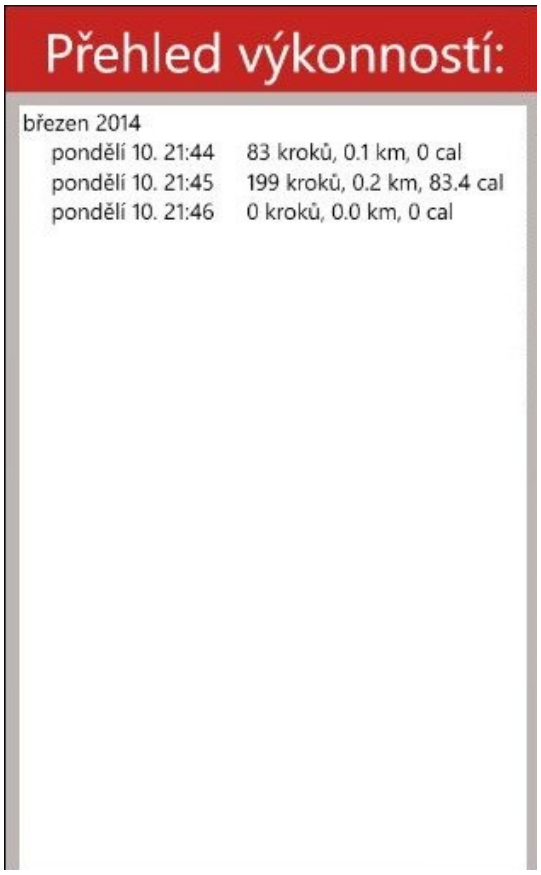
Zdrojový kód 3.4: Změna dat na dlaždici

```
ShellTile.ActiveTiles.First().Update(new StandardTileData()
{
      Title = "StepCounter",
      BackContent = String.Format("{0} kroků \n {1:f2} km \n {2:f1} cal",
      Settings.StepsCount.Value, this.kilometres, this.calories),
      BackgroundImage = new Uri("/Images/walking.png", UriKind.Relative),
      BackBackgroundImage = new Uri("/Images/red.png", UriKind.Relative),
});
```

Obrázek 3.3: Dlaždice krokoměru

3.4 IsolatedStorage

Pro trvalý zápis denních výkonů uživatele byl použit základní typ úložiště *IsolatedStorage* se zápisem do textového souboru. Při zpětném čtení z textového souboru jsou záznamy seřazeny podle měsíce, dne a hodiny od nejnovějších. Před zápisem i čtením dochází ke kontrole, zdali soubor vůbec existuje, aby nedošlo k nestabilitě celé aplikace.



Obrázek 3.4: Grafický výpis z IsolatedStorage

3.5 ApplicationSettings

Pro ukládání uživatelských údajů a zpracovávaných dat byla vytvořena třída *Settings* implementující *ApplicationSetting* úložiště typu *IsolatedStorage*, které umožňuje zpracovávat generické datové typy. Pracuje na principu datového typu *Dictionary*, kdy ke každé hodnotě je přiřazen klíč pro snadné vyhledávání. Každý záznam ve slovníku je určen datovým typem.

Zdrojový kód 3.5: Vytvoření datové struktury ApplicationSettings

```
public static readonly Setting<int> Vaha = new Setting<int>("Vaha", 80);
public static readonly Setting<int> Vyska = new Setting<int>("Vyska", 187);
public static readonly Setting<int> StepsCount = new Setting<int>("StepsCount", 0);
public static readonly Setting<bool> FirstRun = new Setting<bool>("FirstRun", true);
public static readonly Setting<string> PreviousMonth = new
Setting<string>("PreviousMonth","prosinec 2013");
```

4 Aplikace pro běžce

Každodenní výskyt ve stresovém prostředí a práce u počítačů zapřičiňuje tloustnutí a ztrátu kondice. Proto se mezi lidmi začala rozšiřovat obliba běhání po nejbližším okolí. S příchodem nových mobilních systémů a zapracování všemožných druhů senzorů do jediného zařízení se tato zařízení stala každodenním pomocníkem. Pro případ běžeckých a turistických aktivit vznikly aplikace zaznamenávající trasu, uběhnutou vzdálenost, spálené kalorie, tepovou rychlost a mnoho dalších informací a tím umožnily rozšíření trendu běhání i mezi jedince, kteří by se samostatně neodvážili.

Veškerá činnost aplikaceje koncentrována na hlavní obrazovku, kde uživatel vidí údaje o vykonávané činnosti a dokonce zaznamenávání uběhnuté trasy na mapě v reálném čase. Své výsledky může samozřejmě uložit a zobrazit si celkovou uběhnutou trasu. V aplikaci se tedy musí vzít v potaz práce s GPS senzorem, mapami, agenty a zakreslováním tratě pomocí grafických objektů.

4.1 Mapy

Komponenta *Maps* je dalším vylepšením ve verzi Windows Phone, jelikož nově přináší plně vektorové mapy umožňující čistější přechody při změnách velikostí a při 3D transformacích. Mapy mají funkci centrování pozice, nastavení úrovně přiblížení, nastavení úhlu pohledu, zobrazení významných objektů, možnost vykreslení zkratek pro chodce a mnoho dalšího. Ve Windows Phone jsou pro online navigaci použity Bing mapy a pro offline navigaci aplikace třetích stran, příkladem Nokia Here.

Zdrojový kód 4.1: Přidání grafického objektu typu Map (XAML)

```
<maps:Map x:Name="Map" PedestrianFeaturesEnabled="True" LandmarksEnabled="True"
Pitch="53" ZoomLevel="19" />
```

Obrázek 4.1: Použití mapy jako hlavního prvku UI

4.2 Určování polohy a vzdáleností

Pro určování polohy se používá GPS senzor s navázaným družicovým spojením a třída *GeoCoordinateWatcher*, která předává údaje a informace o každé uživatelské změně polohy díky události *PositionChanged*. Pro určování vzdálenosti v metrech se využívá metody *GetDistanceTo* třídy *GeoCoordinate*. [15] Tato metoda přijímá jako parametr objekt třídy *GeoCoordinate*, který má osm atributů:

- **Latitude** (Zeměpisná výška) je typu double, mezi -90 až 90 stupni
- **Longitude** (Zeměpisná délka) je typu double, mezi -180 až 180 stupni
- **Altitude** (Nadmořská výška) je typu double
- **HorizontalAccuracy, VerticalAccuracy** (Přesnost) jsou typu double
- **Course** (Směr) typu double, rozmezí 0 až 360 stupňů
- **Speed** (Rychlost) je typu double
- **IsUnknown** (Neznámý) je typu Boolean (Pokud zařízení nemá povolenu lokalizaci)

Uběhnutá vzdálenost se tedy bude vypočítávat jako součet vzdáleností všech změn poloh od předchozího umístění. Se zavedením časovače třídy *DispatcherTimer* jsem dále schopen určovat rychlosti mezi jednotlivými změnami poloh.

Zdrojový kód 4.2: Vytvoření časovače, zaregistrování metody pro kontrolu změn poloh

```
private DispatcherTimer timer = new DispatcherTimer(); //Instance časovače

//Instance poskytovale lokalizačních údajů
private GeoCoordinateWatcher watcher = new
GeoCoordinateWatcher(GeoPositionAccuracy.High);

private double kilometres;
private double miles;
private double avgSpeed;
private int calories;
private long previousPositionChange;

//Registrování metody pro změnu polohy
watcher.PositionChanged += Watcher_PositionChanged;

private void Watcher_PositionChanged(object sender,
GeoPositionChangedEventArgs<GeoCoordinate> e)
{
      var coordinates = new GeoCoordinate (e.Position.Location.Latitude,
      e.Position.Location.Longitude); //Souřadnice se mění při každé změně

      int count = 0;

      //Pokud již došlo k zakreslení do mapy
      if (line.Path.Count > 0)
      {
            count++;
            var previousPoint = line.Path.Last();//Poslední zakreslená souřadnice
            var distance = coordinates.GetDistanceTo(previousPoint);

            var secondsFromPrevious = (System.Environment.TickCount -
            previousPositionChange)/1000;
            double speed = (distance / secondsFromPrevious) * 3.6;

            this.kilometres += distance / 1000.0;
            this.miles += (distance / 1600);
            this.avgSpeed += speed / count;
            this.calories = (int) this.kilometres * 67;

            AvgSpeed.Text = string.Format("{0:f1}", this.avgSpeed);
            Speed.Text = string.Format("{0:f1} km/h", speed);
            DistancetKm.Text = string.Format("{0:f2} km", this.kilometres);
            DistanceM.Text = string.Format("{0:f2} m", this.miles);
            Calories.Text = string.Format("{0:f0}", this.kilometres * 67);
```

```
            //Natáčení mapy podle aktuální pozice běžce
            PositionHandler handler = new PositionHandler();
            var heading = handler.CalculateBearing(new Position(previousPoint),
            new Position(coordinates));
            Map.SetView(coordinates, Map.ZoomLevel, heading,
            MapAnimationKind.Parabolic);
      }
      else
      {
            Map.Center = coordinates;
      }

      previousPositionChange = System.Environment.TickCount;
}
```

4.3 Zakreslování do mapy

Objekt *Map* je zjednodušeně řečeno vektorové plátno, či objektový kontejner umožňující přidávat jakékoliv grafické objekty. Pro zjednodušení práce jsou v *Map* již předdefinovány prvky *MapPolyLine* a *MapPolygon*. [16] Pro vykreslování běžeckých drah se využívá objektu *MapPolyLine* zakreslujícího se na pozici získanou z objektu *GeoCoordinate* při každé zaznamenané změně uživatelovy polohy.

Zdrojový kód 4.3: Vytvoření grafického objektu pro vyobrazování souřadnic na mapě

```
private MapPolyline line;

public MainPage()

{

      …
      line = new MapPolyline();
      line.StrokeColor = Colors.Red;
      line.StrokeThickness = 20;
      //MapElements je kolekce grafických objektů
      Map.MapElements.Add(line);
}

private void Watcher_PositionChanged(object sender,
GeoPositionChangedEventArgs<GeoCoordinate> e)
{
      …
      //Změna velikosti line na základně nových souřadnic
      line.Path.Add(coordinates);

}
```

Obrázek 4.2: Zakreslení běžené trasy

4.4 Průběh operací v pozadí

V případě běžecké aplikace bude potřeba, aby zaznamenávaní trasy i výpočty byly nepřetržitě vykonávány, ať už při návratu na hlavní plochu nebo při zamčení zařízení. K tomuto účelu nabízí Windows Phone systémového background agenta, přímo určeného pro vykonávání lokalizačních služeb. Tato vlastnost se vykonává skrze kód v WMAppManifest.xml na místě *Tasks*, kde se musí tento agent přidat.

Zdrojový kód 4.4: Přidání systémového agenta (XAML)

```
<Tasks>
      <DefaultTask Name="_default" NavigationPage="MainPage.xaml">
            <BackgroundExecution>
                  <ExecutionType Name="LocationTracking" />
            </BackgroundExecution>
      </DefaultTask>
</Tasks>
```

4.5 LiveTile

V tomto projektu je použita ikonická dlaždice. Dlaždice má pouze přední stranu s ikonou reprezentující aplikaci a aktuálními daty (počet uběhnutých kilometrů, čas a počet spálených kalorií). Je to z toho důvodu, kdy bude mít uživatel zařízení v kapse a bude chtít zkontrolovat uběhnutou vzdálenost, tak vstoupí rovnou na hlavní obrazovku a uvidí data na dlaždici. Nebude tedy muset vyčkávat na detailnější přehled ze zadní části dlaždice, ale veškeré potřebné informace má dostupné ihned.

Zdrojový kód 4.5: Aktualizace dat na dlaždici

```
ShellTile.ActiveTiles.First().Update(new IconicTileData()
{
      Title = "RunNav",
      WideContent1 = string.Format("Vzdálenost: {0:f2} km ", this.kilometres),
      WideContent2 = string.Format("Rychlost: {0:f1} km/h, Spáleno: {1:f0}
      kalorií", speed, this.kilometres * 67),
      WideContent3 = string.Format("Čas: {0}",
      TimeSpan.FromMilliseconds(System.Environment.TickCount -
      time).ToString(@"hh\:mm\:ss")),
});
```

Obrázek 4.3: Dlaždice běžecké aplikace

5 Šipky

Hru s názvem Šipky není potřeba představovat. V aplikaci věnované této tématice jde o to, aby uživatel naházel co nejvíce bodů v šesti hodech. Zároveň se z každé hry vytvářejí statistiky pro možnost motivace v další hře.

Z důvodu realističtějšího vyobrazení hodu šipky jsem se musel zaměřit na animační procesy a manipulaci s nimi. Důležitou části aplikace je pochopení manipulačních gest, a proto je této technice věnována největší část kapitoly. Co se týče uživatelského rozhraní, to je vesměs tvořeno jednoduchými grafickými objekty.

5.1 Dotyková gesta

Dotyková gesta jsou základním prvkem interakce uživatele a zařízení. Gesta jsou definována jako uživatelem započatý pohyb jedním nebo více prsty na dotykové obrazovce. S podporou dotykového vstupu lze vytvořit interaktivní aplikace s mnohem intuitivnějším ovládáním. Pro rozeznání a manipulaci s gesty se využívají události.

Tabulka 5.1: Události spojené s gesty. Zdroj [17]

Událost	Popis
ManipulationStarted	K této události dochází, když uživatel umístí prst, prsty na obrazovku
ManipulationDelta	K této událostí dochází, když uživatele prstem po obrazovce pohybuje
ManipulationCompleted	K této událostí dochází vždy, když se uživatel prstem vzdálí od obrazovky

Tento systém událostí pokrývá mnohé případy manipulace s obrazovkou a nabízí mnohá gesta. S těmito gesty dokáže pracovat většina UIElementů a to díky třídě *GestureListener*, která umožňuje gestům přiřadit funkčnost u daného elementu.

Tabulka 5.2: Metody třídy GestureListener. Zdroj [17]

Typ gesta	Popis
Tap	Prst se dotkne obrazovky a následně ji opouští
DoubleTap	Reprezentuje dvojí poklepání pro potvrzení
Touch & Hold	Prst zůstává položen na obrazovce po určitou časovou dobu
FreeDrag	Prst se dotýká obrazovky a pohybuje se určitým směrem
VerticalDrag	FreeDrag s pohybem nahoru a dolů
HorizontalDrag	FreeDrag s pohybem doleva a doprava
DragComplete	Označuje konec FreeDrag, VerticalDrag, HorizontalDrag gest
Flick	Prst se bez zastavení od daného místa rozjede a v určitém místě opustí obrazovku (Gesto švihnutí)
Pinch & Stretch	Dva prsty simulující pohyb v kruhu a změnu velikosti
PinchComplete	Označuje konec Pinch gesta

5.1.1 Gesto Flick

Gesto *Flick* se dokonale hodí při použití hodu šipky, kdy uživatel položí prst na šipku a švihem kupředu udá vzdálenost a směr dopadu. Důležitým krokem je zpřístupnění gesta pro celou stránku pomocí *GestureListener*, na který je v případě gesta *Flick* zaregistrována vlastní metoda, co se má vykonat při zachycení gesta. [18] *GestureListener* je součástí rozšiřovacího balíčku toolkit, který musí mít vývojář nainstalován, aby se dostal k této funkčnosti.

Zdrojový kód 5.1: Zaregistrování vlastní metody pro vykonání Flick gesta (XAML)

```
<!--Gesta povolena pro celou stranku-->
<toolkit:GestureService.GestureListener>
      <toolkit:GestureListener Flick="GestureListener_Flick"/>
</toolkit:GestureService.GestureListener>
```

Metoda *GestureListener_Flick* udává, co se má vykonat při zachycení gesta *Flick*. V případě této aplikace se musejí zajistit tyto kroky:

1. Zjistit zdali momentálně nedochází k hodu šipky
2. Zajistit kontrolu hozeného počtu šipek, při 0 šipkách začít animaci nové hry
3. Předat data zaznamenaných při vykonání gesta pro vykreslení animací

Instance třídy *FlickGestureEventArgs* vystupujících ve funkci jako parametr třídy, předává hodnoty vlastností potřebných pro určení hodnot, které uživatel gestem vykonal.

- **Angle –** hodnota typu double vyjádřená ve stupních (0-180°)
- **Direction –** objekt typu Orientation, směr tahu
- **HorizontalVelocity –** objekt typu double vyjádřený horizontální rychlostí tahu
- **VericalVelocity -** objekt typu double vyjádřený vertikální rychlostí tahu

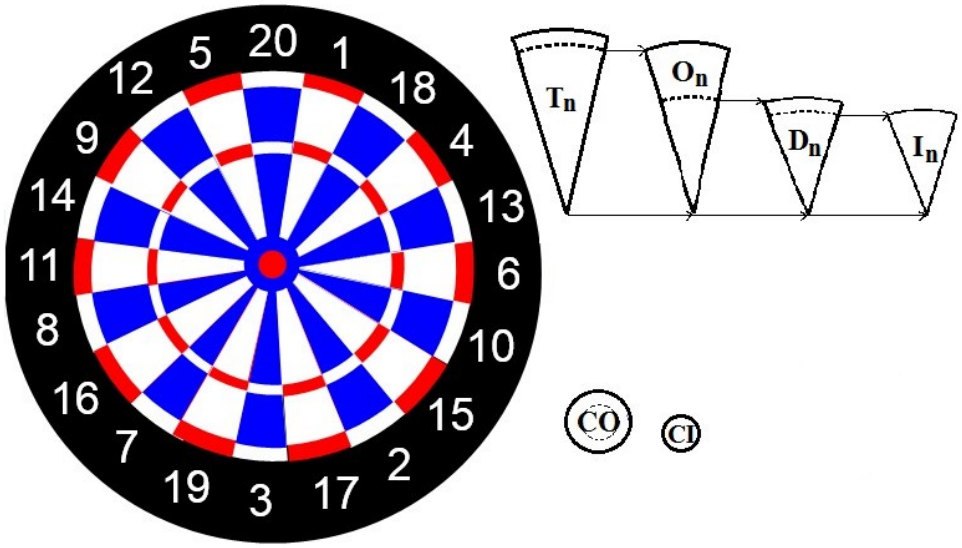
Zdrojový kód 5.2: Metoda vykonávající Flick gesto

```
void GestureListener_Flick(object sender, FlickGestureEventArgs e)
{
	// Podmínka jestli momentálně nedochází k vykonávání animace
	if (this.HozeniSipky.GetCurrentState() == ClockState.Active ||
	this.SpozdeniHodu.GetCurrentState() == ClockState.Active)
	return;

	// Pokud jsou šipky doházeny, dochází k resetování hodnot
	if (this.zbyvajici == 0)
	{
		this.HolesCanvas.Children.Clear();
		this.zbyvajici = 6;
		this.skore = 0;
		this.pocetHozenychBodu = 0;
		this.NavratSipkyZpet.Begin();
	}

	this.DartTransform.TranslateX = 0;
	this.DartTransform.TranslateY = 0;

	//Předání X,Y souřadnic vypočtených z rychlosti tahu jednotlivým animacím os
	this.OsaXAnimace.By = e.HorizontalVelocity / 10;
	this.OsaYAnimace.By = e.VerticalVelocity / 10;

	this.DartImage.Source = this.dartBig;
	this.HozeniSipky.Begin();
}
```

5.2 Uživatelské rozhraní

Úvodní obrazovka sjednocuje veškerou činnost aplikace na jednu stránku. Při tvorbě grafického rozhraní bylo použito několik *Canvas* pláten, na které jsou umístěny všechny grafické a zobrazovací elementy. Také byly využity kontejnery *Storyboard*, které se využívají při animacích. Jedná se o kontejner, do kterého se vkládají objekty, nad nimiž se mají provádět animační efekty.

Hlavní prvek celé aplikace neboli terč, který je složen z 82 samostatných objektů tvaru výseče reprezentujících hrací pole a dalších 20 popisků hozených hodnot. Využity byly také grafické elementy *Ellipse*, *Path* a *TextBlock*. Problémem při tvorbě terče bylo, jak namodelovat objekt tvaru výseče. Pro tento případ .NET Framework nabízí objekt *Path*, který vykresluje soubory souřadnic propojených skrze linie a křivky (princip bézierovy křivky).



Obrázek 5.1: Segmenty tvořící terč (n reprezentuje číslo oddílu)

Zdrojový kód 5.3: Grafické vyobrazení terče (XAML)

```
<Canvas x:Name="BackgroundImage" Margin="0,12,0,0">

      <Ellipse Width="480" Height="480" Fill="Black" />

      <Canvas x:Name="DartboardSegments">
            <Path x:Name="T1" Data="M268.269483095, 61.50827908C287.000166325,
            64.476988515, 305.148564115, 70.37132529,322.042831735,
            78.980320525L240.000027675, 240.000437465z" Fill="Red"
            Canvas.Left="239.417" Canvas.Top="60.925" Stretch="Fill"
            Width="83.209" Height="179.658" />
            … //dalších 79 segmentů terče a 2 středové kruhy
      </Canvas>
      <Canvas>
            <TextBlock Text="20" Canvas.Left="218" Canvas.Top="10"
            FontFamily="Arial" FontSize="40" Foreground="White"/>
            … //dalších 19 popisků
      </Canvas>
</Canvas>
```

Obrázek 5.2: Úvodní sestavení uživatelského rozhraní

5.3 Animace

Výše zmiňovaný kontejner *Storyboard* se využívá pro umisťování objektů, nad kterými se mají vykonávat animace. Umožňuje určit dobu, kdy se má efekt vykonat, jak dlouho má animace trvat a co se má stát po dokončení. Pro rozšíření základních animačních činností se využívají objekty odvozené od typů objektů a vlastností. Všechny tyto animace pracují se společnými metodami pro úpravu cílové vlastnosti. Tedy metody *From*, *To*, *By*, *Duration* neboli odkud, kam, o kolik a doba trvání. [19] Pro použití těchto metod je ovšem nejdůležitější stanovit, nad kterým objektem (*TargetName*) a nad kterou vlastností (*TargetProperty*) objektu se budou změny vykonávat. V tomto případě to bude šipka a vlastnosti pro změnu její polohy na osách X, Y (*TranslateX, TranslateY*) a velikosti (*ScaleX, ScaleY*).

Tabulka 5.3: Základní typy animací. Zdroj [19]

Animace	Popis
DoubleAnimation	Animuje cílovou hodnotu vlastnosti typu double, která se má změnit za daný čas
ColorAnimation	Animuje přechod barev za daný čas
PointAnimation	Animuje posun objektu po přímé linii

V aplikaci jsou animace vesměs pouze na transformaci šipky. Změna velikostí a posun po obrazovce za určitých okolností. Bylo tedy potřeba vyřešit animace, kdy prst opouští obrazovku a dokončuje gesto *Flick* a tím zároveň určuje směr letu a dopadu šipky. Pro uvěřitelnější grafickou animaci je proces mezi hodem a dopadem šipky rozdělen na tři samostatné přechody (hod, let, dopad). Při každém z těchto přechodů se mění rozměry a také rychlost letu šipky, aby to vypadlo, že šipka směrem od nás letí pomalu, ale později zrychluje.

Obrázek 5.3: Šipky použité pro animaci (počáteční, za letu, dopad)

Zdrojový kód 5.4: Vytvoření objektu reprezentujícího šipku a animací nad ním (XAML)

```
<!–Šipka s TranslateY property pro počáteční pozici-->
<Image x:Name="DartImage" Canvas.Top="552" Canvas.Left="190" Width="100">
	<Image.RenderTransform>
		<CompositeTransform x:Name="DartTransform" TranslateY="130"/>
	</Image.RenderTransform>
</Image>

<!--Storyboard jenž má na starost animaci hodu od uživatele k terči -->
<Storyboard x:Name="HozeniSipky" Completed="HozeniSipky_Completed">

	<!-- Animace určující tvar letu šipky -->
	<DoubleAnimation x:Name="OsaXAnimace" Duration="0:0:.3"
	Storyboard.TargetName="DartTransform"
	Storyboard.TargetProperty="TranslateX">
		<DoubleAnimation.EasingFunction>
			<CircleEase/>
		</DoubleAnimation.EasingFunction>
	</DoubleAnimation>

	<DoubleAnimation x:Name="OsaYAnimace" Duration="0:0:.3"
	Storyboard.TargetName="DartTransform"
	Storyboard.TargetProperty="TranslateY">
		<DoubleAnimation.EasingFunction>
			<CircleEase/>
		</DoubleAnimation.EasingFunction>
	</DoubleAnimation>

	<!-- Animace určující velikost šipky v jednotlivých fázích letu -->
	<DoubleAnimation x:Name="velikostSipkyXAnimace" Duration="0:0:.3"
	Storyboard.TargetName="DartTransform" Storyboard.TargetProperty="ScaleX"
	By="1.5" AutoReverse="True">
		<DoubleAnimation.EasingFunction>
			<CircleEase/>
		</DoubleAnimation.EasingFunction>
	</DoubleAnimation>

	<DoubleAnimation x:Name="velikostSipkyXAnimace" Duration="0:0:.3"
	Storyboard.TargetName="DartTransform" Storyboard.TargetProperty="ScaleY"
	By="3" AutoReverse="True">
		<DoubleAnimation.EasingFunction>
			<CircleEase/>
		</DoubleAnimation.EasingFunction>
	</DoubleAnimation>
</Storyboard>
```

Zdrojový kód 5.5: Logika vykonávající události po dopadu šipky

```
void HozeniSipky_Completed(object sender, EventArgs e)

{
	this.DartImage.Source = this.dartSmall;

	// Vytvoření tečky reprezentujícího dopad šipky na terč
	Point umisteniDirky = new Point(Canvas.GetLeft(this.DartImage) +
	this.DartTransform.TranslateX + this.DartImage.Width / 2 - 10,
	Canvas.GetTop(this.DartImage) + this.DartTransform.TranslateY + 43);

	Ellipse dira = new Ellipse { Fill = new SolidColorBrush(Colors.Black),
	Width = 5, Height = 5 };

	Canvas.SetLeft(dira, umisteniDirky.X - dira.Width / 2);
	Canvas.SetTop(dira, umisteniDirky.Y - dira.Height / 2 - 43);
	this.HolesCanvas.Children.Add(dira);

	// Výpočet hodu na místě dopadu, odečtení počtu hodů, spuštění animace sipky
	pocetHozenychBodu = VypocitejBody(umisteniDirky);
	this.skore += pocetHozenychBodu;
	this.zbyvajici--;
	this.SpozdeniHodu.Begin();

	UpdateSkore();

	this.PocetHozenychBoduTextBlock.Text = pocetHozenychBodu.ToString();

	// Pokud jsou dané šipky vyházeny, dochází k vynulování a nastavení nové hry
	if(this.zbyvajici == 0)
	{
		this.prumSkore.Value = ((this.prumSkore.Value *
		this.pocetHer.Value) + this.skore) / (this.pocetHer.Value + 1);
		this.pocetHer.Value++;

		 GameOver.Begin();
		 this.HolesCanvas.Children.Clear();
		 this.GameOverTB.Visibility = Visibility.Visible;
		 this.NewGameTB.Visibility = Visibility.Visible;

		 if(this.skore > this.nejSkore.Value)
		 {
			this.nejSkore.Value = this.skore;
		 }

		this.PocetHozenychBoduTextBlock.Text = "0";
		this.ScoreTextBlock.Text = "0";
		this.AvgScoreTextBlock.Text = this.prumSkore.Value.ToString("#.##");
		this.BestScoreTextBlock.Text = this.nejSkore.Value.ToString();

		this.UkonceniHry.Begin();
	}
}
```

6 Náročnost a efektivita aplikací

Každý začínající vývojář se v první řadě rozhoduje, prostřednictvím které platformy bude své aplikace vydávat na trh. Proto jsem provedl základní srovnání platforem, které momentálně drží největší procento tržního podílu (Android, iOS, Windows Phone). Při porovnávání půjde nejen o celistvý přehled prostředků potřebných pro vývoj na jednotlivých platformách, ale i o pozitivech a negativech jednotlivých systémů, které se také musí brát v potaz.

6.1 iOS

V roce 2007 společnost Apple poprvé představila svůj mobilní operační systém založený na Unixovém jádře. Již od samého počátku je systém navržen jako uzavřená platforma s přesně danou hardwarovou specifikací, což šetří neuvěřitelně mnoho práce optimalizováním aplikací. Další nespornou výhodou iOS je, že vychází na třech typech zařízení ročně (tablet, mobil, multimediální zařízení) a tudíž není ekosystém narušen všemožnými hardwarovými specifikacemi. Díky tomuto může Apple podporovat svá zařízení až 5 let.

Co se týče tržního podílu, byl iOS několik let na špici, ovšem dokud se situace v roce 2011 neotočila ve prospěch android. Pro rok 2013 měl systém podíl 15,5%, což činí 153,4 miliónů mobilních zařízení. [20] Ohledně zákazníků se říká, že uživatelé iOS si rádi zaplatí za exkluzivitu a reálně to tak i funguje.

Jasnou nevýhodou iOS je její uzavřenost, tudíž si zařízení různých platforem nemohou vyměňovat data skrze bezdrátová spojení a ani standartní propojení s počítači nefunguje. Dále nutnost využívání Apple software pro synchronizace a zálohy dat.

6.2 Android

Společnost Google v roce 2007 (vývoj ohlášen již 2005) přispěchala se svým open source systémem nazvaným Android. Je založen na linuxovém kernelu, tudíž umožňuje běh systému na různých druzích hardware. Díky tomu se začali prodávat mnohem levnější telefony, ale přiznejme si, že stabilita a funkčnost na takových zařízeních nebyla na vysoké úrovni. S každou další iterací systému se tento problém postupně vymazával a umožnil Androidu stát na nejvyšším stupínku tržního podílu s mobilními zařízeními. Pro rok 2013 byl podíl systému 78,9%, což činí 781,2 miliónů mobilních zařízení. [20]

Hlavním nedostatkem systému Android je roztříštěnost verzí systému a použití nejrůznějších druhů hardware. Tento nedostatek představuje veliký problém pro vývojáře a cílení aplikací pro různá rozlišení, výkony a různé verze systému. Dalším zásadním nedostatkem spojeným s velkým počtem vyrobených druhů zařízení, že je výrobci nepodporují déle než rok.

6.3 Windows Phone

Koncem roku 2010 byla světu představena platforma Windows Phone a to ve verzi 7 založeném na jádře Windows CE. Platforma si nezískala moc velký obdiv, jelikož přišla s mnoha chybějícími funkcemi a tudíž se nemohla rovnat s konkurencí. Na to Microsoft zareagoval a ve velmi krátké době vydal aktualizované verze 7.5 a 7.8 kde se snažil dohnat nedostatky. V době vydání verze 7.8 Microsoft přehodnotil své cíle a vytvořil nový systém ve verzi 8, který není zpětně kompatibilní se staršími verzemi, jelikož je založen na novém jádru Windows NT a tím si znepřátelil mnoho zákazníků. Ovšem díky tomuto kroku se platforma začala konečně řádně rozvíjet a přinesla všechna potřebná vylepšení, která dodnes umožňovala konkurenci technologicky utíkat.

Windows Phone bere výhody obou konkurenčních platforem a spojuje je v jedno řešení. Hlavním kladem je daná hardwarová specifikace a díky tomu je systém velice odladěný, stabilní a tedy i velice svižný. Navíc Microsoft slibuje, že všechna zařízení budou mít minimálně 18 měsíční podporu. Momentálním hlavním nedostatkem je počet aplikací na Store, který činí něco málo přes 200 000 aplikací, což je v porovnání s konkurencí velmi málo. Ovšem pro mnohé vývojáře to může být výhoda, jelikož market ještě není přeplněn a lidé stále čekají na skvělé aplikace.

Ohledně tržního podílu měl Windows Phone pro rok 2013 3,6 %, což činí 35,7 miliónů mobilních zařízení. [20] Systém je ovšem na rychlém vzestupu a podle analytiků se předpokládá, že v roce 2017 zaujme druhou příčku nejpoužívanějších systémů.

Tabulka 6.1: Přehledné srovnání platforem z vývojářského hlediska

	iOS	Android	WP8
Podporované jazyky	C/C++, Objective-C	Java, C/C++	C#,VB,C/C++
SDK	iPhone SDK	Android SDK	WP8 SDK
Studio	xCode	Eclipse, Netbeans	Visual Studio Express
Systém pro vývoj	Mac OS	Windows, Mac, Linux	Windows 8
Emulátor	Ano, svižný	Ano, velice pomalý	Ano, svižný
Obchod	App Store	Google play	Store

Počet aplikací	1 000 000 +	1 100 000 +	200 000 +
Schválení aplikace	2 týdny	Žádná	Týden
Vydání aplikace	Free, paid, trial	Free, paid	Free, paid, trial
Vývojářský poplatek	2000 Kč ročně	500 Kč	0, 357 Kč ročně
Cena nástrojů	100 Kč	zdarma	zdarma
Daň z prodeje	30 %	30 %	30 %
Vyplácení (podmínka)	Měsíčně při 3000 Kč	Měsíčně při 20 Kč	Při 4000 Kč
Testovací zařízení	9 990 – 25 000 Kč	3 500 – 20 000 Kč	3 500 – 15 000 Kč

6.4 Porovnání rychlostí vývoje

Nejjednodušším poukazatelným příkladem rychlosti vývoje je porovnání dob pro vytvoření standartního projektu výuky pro začátečníky s názvem Hello World na jednotlivých platformách. Jedná se o program s tlačítkem vyvolávající změnu textu v elementu pro jeho zobrazení. iOS 2.8 minuty, Android 3 minuty, WP 40 sekund. [21] To je ovšem zapřičiněno tím, že Windows Phone má nejrychlejší návrh UI, jelikož všechny objekty jsou jednoznačně reprezentovány pomocí *x:name* direktivy a zobrazované hodnoty se dají jednoduše přepsat v jediném řádku XAML kódu.

Pro přesnější představu proběhl v roce 2013 celosvětový průzkum mezi vývojáři mnoha studií a došlo se k závěru, že nejrychleji se vyvíjí pro Android, pro který hlasovalo 40,5 % respondentů a těsně v závěsu iOS 36,2 % a na posledním místě Windows Phone s 34 %. [22] Ovšem musíme brát v úvahu, že tento výsledek je ovlivněn tím, že majorita vývojářů se specifikuje pouze na Android a iOS kvůli momentálnímu tržnímu podílu. Tudíž jakmile se změní postavení na trhu, dojde i k přeorientování mnohých studií.

6.5 Multiplatformní vývoj aplikací

V dnešní době kdy jsou úspěchy vývojáře hodnoceny počtem stažení a tedy i následným ziskem si vývojář musí dobře rozmyslet na jaké zákazníky a jakou platformu bude své aplikace cílit. Nejlogičtějším krokem je napsat aplikaci na všechny platformy a tedy cílit na největší spektrum uživatelů. Ovšem když už se vývojář rozhodne vytvořit aplikaci pro Windows Phone, tak jak ji může nejjednodušeji převést pro ostatní systémy nebo zpětně jak převést aplikaci pro Windows Phone?

6.5.1 Microsoft API mapping tool

Ve snaze o přilákání co největšího počtu vývojářů začal Microsoft pracovat na návodech a API pro pomoc při převodu aplikací jak z iOS, tak z Androidu. [23] API se neustále rozšiřují a přibývá také mnoho příkladů. Pro vývojáře je tedy mnohem jednodušší si najít ekvivalenty použité třídy, než studovat platformu od úplných základů.

6.5.2 Projekt Mono/Xamarin

Mono je všestranný open-source projekt společnosti Novell, dnes již nazývaný Xamarin, jehož cílem je přinést implementaci .NET frameworku (v poslední verzi .NET framework 4.0) pro různé platformy. [24] Což se opravdu povedlo a Mono je podporován na operačních systémech Windows, Linux, Unix, Mac, Solaris, BSD, Android, iOS a dokonce i na herních konzolích a to včetně možnosti kompilace aplikací pro jednotlivé platformy. Umožňuje .NET/C# vývojářům psát aplikace jak jsou zvyklí a zároveň jim otevírá dveře na trhy s mobilními aplikacemi.

Doporučovanou technikou návrhu aplikace pro co nejjednodušší převod mezi mobilními platformami je použití vzoru MVVM nebo knihovny PCL. Xamarin dokonce nabízí skener mobility, který dokáže zjistit množství kódu ze stávající aplikace, které je možno znovu použít. [24] Tento znovupoužitelný kód umístíme do centrální knihovny přístupné pro všechny platformy a připojíme UI pro daný systém.

6.5.3 Projekt PhoneGap

Open-source framework vyvinutý firmou Nitobi, kterou v roce 2011 skoupila společnost Adobe Systems. Framework umožňuje vývojářům psát aplikace pro mobilní zařízení pomocí jazyků JavaScript, HTML5 a CSS3. Principem funkčnosti je použití *WebView* prvku na plné velikosti displeje, který nahrazuje standartní uživatelské rozhranní. Komunikace aplikace a operačního systému probíhá pomocí pluginů a tudíž je vyžadováno připojení zařízení k internetu. Framework podporuje většinu mobilních platforem.[25]

6.5.4 Projekt MoSync

Jde o pokročilý open-source nástroj pro multiplatformní vývoj aplikací, který podporuje nejpoužívanější knihovny a technologie potřebných pro vývoj na většině platforem. Podporuje jazyky C/C++, PHP, Python, Ruby, JavaScript a Java. MoSync dokonce umožňuje navrhovat nativní uživatelské rozhranní v jazyce jednotlivých platforem. Jako vývojové studio je použito Eclipse. [25]

6.5.5 **Projekt Whoop**

Kontroverzní nástroj Whoop je jediný, který je určen nejen vývojářům, ale i lidem neznalých programování. Jedná se o *WYSIWYG* editor, kde se prostředí tvoří pomocí přetahování, umisťování a propojování grafických elementů. Výsledné aplikace se dají exportovat do mnoha formátů, podle cílené platformy.

7 Budoucnost platformy Windows Phone

Windows Phone 8 konečně dohnal technologicky své největší konkurenty na platformách Android a iOS. Ovšem z důvodu malého počtu výrobců a tudíž i malého počtu možných konkurujících zařízení je tržní situace pro Microsoft nelichotící. Z Evropského hlediska Windows Phone ovládá pouze 10 % a světově pouze necelé 4 % trhu (Pro zajímavost v tuzemském prostředí je to druhá nejprodávanější mobilní platforma). Nastalý problém připisují vysoké startovní ceně zařízení, která odrazuje potencionální kupce. Díky tomu se mnohokrát stalo, že Nokia slevnila svá zařízení po měsíci prodeje i o 30 % a do půl roku prodeje zařízení dokonce i o 60 %, aby se prodaly cílené kvóty produktů.

Tabulka 7.1: Podíly na trhu s mobilními systémy. Zdroj [19]

Systém	Q4 2012	2012 celkově	Q4 2013	2013 celkově
Android	70,3	68,8	78,4	78,9
iOS	22,0	19,4	17,6	15,5
Windows Phone	**2,7**	**2,7**	**3,2**	**3,6**
Ostatní	5,0	9,1	0,7	2,0
Souhrn	**100**	**100**	**100**	**100**

Dalším problémem WP platformy je minimální zájem vývojářů, a to z nedostatečného počtu potencionálních zákazníků a hlavně těch platících. I samotný Google odmítá své aplikace vyvíjet pro tuto platformu. [26] Tato situace se ovšem může změnit, protože Microsoft zvažuje zrušit licenční poplatek a koupil také celou mobilní divizi Nokie pod záminkou výroby vlastních telefonů. Momentálně ovšem spoléhá na stávající nabídku WP zařízení od Nokie s názvem Lumia a také na sponzorování velkých výrobců zaměřujících se více na open source android pro výrobu několika modelů.

Microsoft má ovšem se všemi svými systémy veliké plány. Ve verzi 8.1 jsou ještě verze systémů Windows rozděleny na stolní, mobilní, tabletovou a konzolovou. Na konferenci Worldwide Partner Conference bylo řečeno, že s updatem 8.2 má dojít ke sjednocení mobilních systémů RT a WP a s verzí 9 dokonce se stolními Windows. [27] To značí, že již Store nebude existovat odděleně pro jednotlivé verze systémů, ale vytvoří pouze jediný. Vývojáři tedy budou moci publikovat jednu aplikaci na více zařízení bez větších zásahů a úprav kódu.

7.1 Co přinese Windows Phone 8.1

V době psaní této práce byl na Microsoft konferenci Build oficiálně představen WP 8.1 s kódovým označením ‘Blue‘. Došlo také k uvolnění nového SDK [28] a díky tomu na povrch vypluli kompletní informace o nadcházejících novinkách, které systém přináší:

- Cortana – hlasová asistentka
- Notifikační centrum – centralizovaná zpráva, notifikace ze sociálních sítí
- Dual SIM – instalace do nových zařízení
- Podpora větších úhlopříček (7”), kvůli sloučení s Windows RT
- Instalace aplikací a her na paměťovou kartu, svázáno pouze s daným zařízením
- Uživatel si může nastavit libovolné pozadí na dlaždicích i na ploše
- Internet Explorer 11 – plná podpora HTML5 videí, stahování libovolných souborů, správce hesel, 6 záložek najednou, synchronizace záložek s Windows 8
- Softwarová tlačítka (přímo na displeji)
- Podpora nových procesorů a chipsetů
- Noví OEM partneři
- Detailní monitorování spotřeby
- Swype klávesnice (tahová)
- Fotoaparátu přibude možnost vytvořit sérii snímků
- Filtrování obsahu
- Možnost změny cílových adresářů pro ukládání
- Sdílení Wi-Fi sítě s libovolnými kontakty
- Automatické aktualizací aplikací, možnost pouze přes Wi-Fi
- Odemčení telefonu pomocí dvojitého poklepání na displej, uspání stejně
- Screenshot obrazovky
- Synchronizace aplikací
- Změna výchozího klienta pro sms, fotoaparát
- Vylepšený multitasking, tlačítko zpět nevypíná aplikaci, ale uspává. Přehled spuštěných aplikací, které mohou být ukončeny stažením směrem dolů
- Zobrazení obsahu displeje na nedalekých zařízeních (kabel, bezdrátově)
- Přidána rozlišení (384x640, 540x960, 1440x2560)
- Podpora PlayTo, stereoskopické 3D, Wi-Fi Direct
- Dostupnost pro veškerá Windows Phone 8 zařízení
- Možnost odebrat integraci sociálních sítí

Závěr

V této práci jsem podal základní přehled o možnostech a specifikacích mobilního operačního systému Windows Phone 8 a také o nástrojích, technologiích a nárocích potřebných pro vývoj na této platformě. Dále jsem představil základní mechanismy využívané při vývoji aplikací, které jsou také důležité pro pochopení v návaznosti na následující části práce.

Podstatná část práce je věnována realizaci aplikací, na kterých jsem demonstroval práci se základními funkčními prvky zařízení, jako jsou Live Tile, senzorová čtení GPS a akcelerometru, lokalizační služba, čtení a zápis datových zdrojů, grafické objekty, animace a dotyková gesta.

V závěrečné části jsem zhodnotil možnosti systému v porovnání s ostatními konkurenčními platformami na trhu z pohledu vývojáře, nároků, efektivity a náročnosti vývoje. Následně jsem se věnoval převoditelnosti aplikace napříč platformami a možnosti nativního vývoje. Také jsem se zmínil o přítomnosti a nejbližší budoucnosti Windows Phone z pohledu tržní konkurenceschopnosti a technologického vývoje.

Doufám, že práce motivuje vývojáře k tvorbě aplikací pro tuto platformu. Windows Phone je velice mladý systém, kterému ještě nebyla věnována dostatečná pozornost, ale má předpoklad pro zářnou budoucnost.

Použitá literatura

[1] *OS Windows Mobile/Phone: strmá cesta historií* [online]. 2011 [cit. 2014-02-24]. Dostupné z: http://www.cnews.cz/os-windows-mobilephone-strma-cesta-historii

[2] *Windows Phone 7: oficiální představená a první dojmy* [online]. 2010 [cit. 2014-02-24]. Dostupné z: http://mobilenet.cz/clanky/windows-phone-7-oficiln-pedstaven-a-prvn-dojmy-5983

[3] *Optimizing battery consumption of Windows Phone Applications* [online]. 2013 [cit. 2014-02-24]. Dostupné z: http://blogs.windows.com/windows_phone/b/wpdev/archive/2013/01/17/optimizing-battery-consumption-of-windows-phone-applications.aspx

[4] *Windows Phone 8 minimal hardware requirements* [online]. 2012 [cit. 2014-02-24]. Dostupné z: http://wmpoweruser.com/minimum-hardware-requirements-for-windows-phone-8-revealed/

[5] *Hyper-V*. 2013 [cit. 2014-03-07]. Dostupné z: http://cs.wikipedia.org/wiki/Hyper-V

[6] *SLAT*. 2013 [cit. 2014-03-07]. Dostupné z: http://cs.wikipedia.org/wiki/SLAT

[7] *APP HUB* [online]. 2010 [cit. 2014-02-24]. Dostupné z: http://create.msdn.com

[8] *Windows Phone 8 and Windows 8 platform comparison* [online]. 2014 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/ develop.aspx

[9] *Portable Class Library* [online]. 2012 [cit. 2014-02-24]. Dostupné z: http://blogs.msdn.com/b/vyvojari/archive/2012/07/18/portable-libraries-jedna-knihovna-pro-v-echny-platformy.aspx

[10] *App activation and deactivation for Windows Phone* [online]. 2014 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff817008(v=vs.105).aspx

[11] *Background agents for Windows Phone* [online]. 2014 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202942(v=vs.105).aspx

[12] *Tiles for Windows Phone* [online]. 2014 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202948(v=vs.105).aspx

[13] CHUVYROV, Henry Lee and Eugene, Karli. *Beginning Windows Phone 7 development: building Windows Phone applications using Silverlight and XNA*, 2011. ISBN 978-143-0235-972

[14] *Data from the accelerometer for Windows Phone* [online]. 2014 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff431810 (v=vs.105).aspx

[15] *GeoCoordinateWatcher* [online] 2012 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/cs-cz/library/system.device.location.geocoordinatewatcher(v=vs.110).aspx

[16] *Maps and navigation for Windows Phone 8* [online]. 2014 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj207045(v=vs.105).aspx

[17] *Gesture support for Windows Phone* [online]. 2014 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff967546(v=vs.105).aspx

[18] *Building touch interfaces fo Windows Phone* [online]. 2012 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff967546(v=vs.105).aspx

[19] *Quickstart: Animations for Windows Phone* [online]. 2014 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj206955(v=vs.105).aspx

[20] *Mobile platforms market share 2013* [online]. 2014 [cit. 2014-02-24]. Dostupné z: http://www.gartner.com/newsroom/id/2665715

[21] *Developer Efficiency: XCode, Eclipse, Visual Studio* [online]. 2011 [cit. 2014-03-07]. Dostupné z: http://www.youtube.com/watch?v=OF5mGoKcm80

[22] *6000 mobile developers: Android most popular, iOS most profitable, Windows Phone most 'next'* [online]. 2011 [cit. 2014-03-07]. Dostupné z: http://www.youtube.com/watch?v=OF5mGoKcm80

[23] *Port your application* [online]. 2011 [cit. 2014-02-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windows/apps/dn436165

[24] *Introduction to Xamarin* [online]. 2012 [cit. 2014-02-24]. Dostupné z: http://www.slideshare.net/alexrhack/introduction-to-xamarin-31398436

[25] *Cross platform development* [online]. 2013 [cit. 2014-02-24]. Dostupné z: mobiledevices.about.com/od/mobileappbasics/tp/Top-5-Tools-Multi-Platform-Mobile-App.htm

[26] *Google: Vyvíjet pro Windows Phone nebo Windows 8 se nám zatím nevyplatí* [online]. 2012 [cit. 2014-03-07]. Dostupné z: http://smartmania.cz/bleskovky/google-vyvijet-pro-windows-phone-nebo-windows-8-se-nam-zatim-nevyplati-3919?sc=1

[27] *Microsoft codename 'Threshold': The next major Windows wave takes shape* [online]. 2012 [cit. 2014-03-07]. Dostupné z: http://www.zdnet.com/microsoft-codename-threshold-the-next-major-windows-wave-takes-shape-7000023832/

[28] *Windows Phone Blue* [online]. 2014 [cit.2014-03-10]. Dostupné z: http://wmmania.cz/clanky/obecne/windows-phone-blue-prehled-novinek

[29] *Windows Phone: jak začít s vývojem* [online]. 2012 [cit.2014-03-10]. Dostupné z: http://wp7.garvis.cz/tema-zaklady.html

Seznam zdrojových kódů

Seznam tabulek

Seznam obrázků

Seznam příloh

Příloha 1 - Adresářová struktura přiloženého CD

TextBc – elektronická verze bakalářské práce

Aplikace – obsahuje tři aplikace (krokoměr, aplikace pro běžce, šipky)

Prilohy

Images – obrázky využité v rámci aplikací

Texty – podpůrné texty využité při vývoji

Příloha 2 – Publikace mobilní aplikace na Store

Pokud vývojář již odladil a otestoval svou aplikaci podle představ, tak je načase přistoupit k procesu zveřejnění aplikace na Store. S tím ovšem souvisí nutnost podstoupit absolvování publikačního a schvalovacího procesu, díky němuž koncový uživatel získává jistotu o bezpečnosti aplikace. Každá aplikace ve Store prošla a tedy splňuje sady požadavků, které jsou vyžadovány. Po ověření ze strany Microsoftu a splnění požadavků dojde k certifikaci, podepsání a zveřejnění aplikace uživatelům. [29]

Příprava aplikace

Před publikací je potřeba splnit:

- **Ikona aplikace** – standartní ikona aplikace, která bude zobrazena v seznamu aplikací. Ikona má rozměry 62x62 pixelů a formát PNG kvůli průhlednosti
- **Ikona na hlavní obrazovce** – ikona využita při připíchnutí aplikace na hlavní plochu, součást automaticky generovaného Live Tile. Rozměry 173x173 pixelů a formát PNG
- **Startovací obrázek (Splash Screen)** – aplikace se po určitou dobu spouští a proto se využívá úvodní obrazovky aplikace. Rozměry podle rozlišení displeje zařízení (rozlišení 1280x768 pixelů si systém automaticky upraví sám, případně 800x480 pixelů)
- **Manifest aplikace** – soubor WMAppManifest.xml z adresáře vytvořené aplikace. Manifest obsahuje informace o autorovi, aktuální verzi, použitých obrázcích. Obsahuje také ProductID, který slouží pro jednoznačnou identifikaci aplikace ve Store

Při přípravě aplikace na publikaci je také důležité nastavit lokalizaci a jazykovou verzi aplikace. Windows Phone umožňuje aplikace lokalizovat pro více jazyků zároveň. Čeština plně podporována a lokalizaci nastavíme v projektu v záložce *Properties > Application > Assembly Information…*

Dále je také potřeba vytvořit XAP balíček aplikace, který se zkompiluje v režimu *Release* ve Visual Studiu. Výsledný soubor najdete v adresáři *Bin/Release/NazevAplikace.xap*

Doplňkové zdroje pro publikaci

Pokud jste dokončili všechny výše zmiňované kroky, je potřeba ještě připravit několik vizuálních prvků, které budou reprezentovat aplikaci přímo ve Store.

- **Malý obrázek pro mobilní Store** – formát PNG s rozměrem 99x99 pixelů
- **Velký obrázek pro PC Store** – formát PNG s rozměrem 200x200 pixelů
- **Snímky z aplikace** – snímky obrazovky z aplikace při jejím reálném použití. Formát PNG v rozměrech rozlišení zařízení (480x800 pixelů nativní). Musí být nahrán minimálně jeden a maximálně osm snímků.

Publikační proces

Postup pro publikaci aplikace:

1. Přihlášení pod registrovaným vývojářským účtem [7] a zvolení Submit a new app!
2. Prvním krokem je nahrání XAP balíčku a vyplnění údajů o aplikaci (jméno aplikace, verze). Stránka také obsahuje možnost vydání aplikace na veřejný či testovací Store. Druhá možnost představuje možnost přístupu vybraným testerům
3. V dalším kroku probíhá popis aplikace s klíčovými slovy
4. Dále se určí cena aplikace (zdarma, určená cena). Nastavuje se zde i lokalita, pro kterou bude aplikace vydána
5. Posledním krokem je zvolení možnosti publikování (ručně, automaticky)

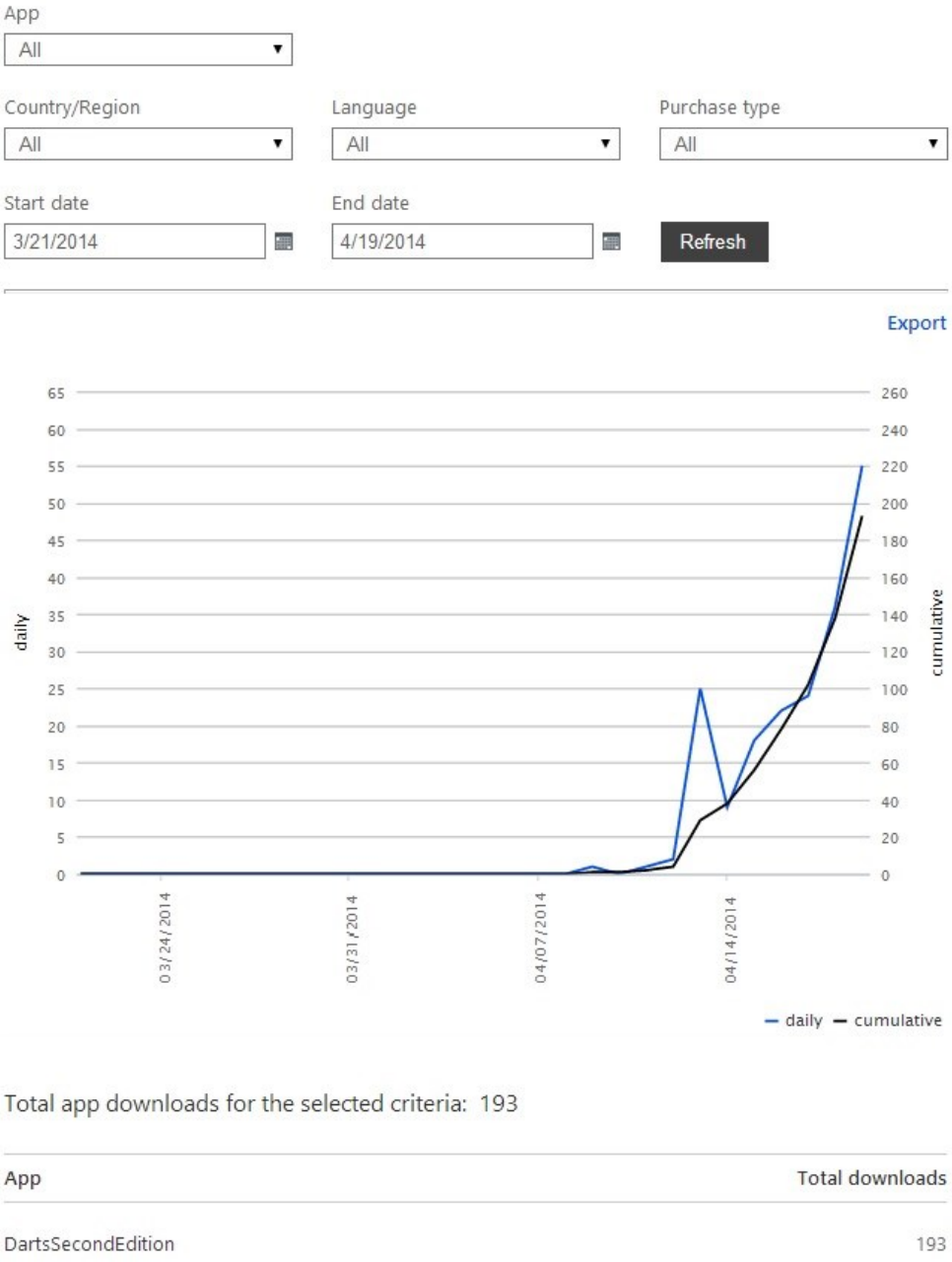
Po dokončení jednotlivých kroků je již zbytek schvalovacího procesu v rukou Microsoftu. Aplikace projde touto sadou testovacích podmínek.

- **Požadavky na aplikaci** - obecně vyžadované vlastnosti a funkce aplikace, a to jak s ohledem na technické řešení, tak na obchodní model a obsah aplikací
- **Požadavky na obsah** - požadavky směřující k samotnému obsahu a účelu aplikací. Vymezují pravidla pro používání ochranných známek, vulgárních výrazů, skryté reklamy či reklamy.
- **Požadavky na publikaci aplikací** - požadavky úzce spojené se samotným procesem publikace a šíření aplikace. Zabývají se především obsahem XAP balíku a doplňkových ikon, snímků obrazovky
- **Požadavky technického řešení** – požadavky zabývající se způsobem využívání zdrojů (maximální využitá paměť, čas spuštění aplikace), interakce s uživatelem (konzistentní funkčnost tlačítka zpět, jazyková lokalizace)
- **Doplňkové požadavky** - další dodatečné požadavky, např. informování uživatele o požadavcích aplikace

Projde-li aplikace úspěšně procesem certifikace, bude vývojář informován e-mailem a aplikace se buď umístí na Store nebo se počká na ruční vydání. Na vývojářském účtu je také možnost sledovat statistiky stahovatelnosti a oblíbenosti aplikace.

Pokud se certifikace nezdaří, dostane vývojář PDF, ve kterém budou vypsány chyby a doporučení pro příští publikaci.

Přehled stahovanosti publikované aplikace



Total app downloads for the selected criteria: 193

App	Total downloads
DartsSecondEdition	193

Marketing

Publikovaním aplikace ještě práce nekončí. Pokud vývojář očekává, že bude aplikace úspěšná a bude se umisťovat na vyšších žebříčcích stahovanosti, musí pro to něco udělat. Hlavní prvek úspěšnosti aplikace je její povědomí mezi lidmi, neboli reklama.

Reklamy jsou dvojího typu finančních náročností:

Zdarma – sociální sítě (facebook, twitter)

Placené – reklamní bannery na dostatečně navštěvovaných webech, odkazy v médiích